Explaining Machine Learning Predictions: Rationales and Effective Modifications

by

Sudhanshu Nath Mishra

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2018

Author ... Signature redacted ....
Department of Electrical Engineering and Computer Science
August 8, 2018

Certified by...... Signature redacted ....
Randall Davis
Professor, Department of Electrical Engineering & Computer Science
Thesis Supervisor

Certified by........... Signature redacted ....
Andrew W. Lo
Charles E. and Susan T. Harris Professor, Sloan School of Management
Thesis Supervisor

Accepted by........... Signature redacted ....
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Explaining Machine Learning Predictions: Rationales and Effective Modifications

by

Sudhanshu Nath Mishra

Submitted to the Department of Electrical Engineering and Computer Science
on August 8, 2018, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Deep learning models have demonstrated unprecedented accuracy in wide-ranging tasks such as object and speech recognition. These models can outperform techniques traditionally used in credit risk modeling like logistic regression. However, deep learning models operate as black-boxes, which can limit their use and impact. Regulation mandates that a lender must be able to disclose up to four factors that adversely affected a rejected credit applicant. But we argue that knowing why an applicant is turned down is not enough. An applicant would also want actionable advice that can enable them to reach a favorable classification. Our research thus focuses on both the desire to explain why a machine learning model predicted the classification it did and to find small changes to an input point that can reverse its classification. In this thesis, we evaluate two variants of LIME, a local model-approximation technique and use them in a generate and test algorithm to produce mathematically-effective modifications. We demonstrate that such modifications may not be pragmatically-useful and show how numerical analyses can be supplemented with domain knowledge to generate explanations that are of pragmatic utility. Our work can help accelerate the adoption of deep learning in domains that would benefit from interpreting machine learning predictions.

Thesis Supervisor: Randall Davis
Title: Professor, Department of Electrical Engineering & Computer Science

Thesis Supervisor: Andrew W. Lo
Title: Charles E. and Susan T. Harris Professor, Sloan School of Management

Acknowledgments

My journey to MIT feels nothing short of a dream come true. Growing up in Bangalore, India, I always aspired for excellence. I believed that one day, I would graduate from one of the finest academic institutions in the world and would then go on to become a professional basketball player in the NBA. As you can imagine, only one part of that dream has and ever will come true. What I could have never foreseen, however, is the unlikely series of steps that have led me to my milestone.

I took an unexpected gap year after high school and was accepted to MIT on my second attempt. I happened to complete my undergraduate requirements in the first three years at MIT and thus had the opportunity to pursue my Masters in the fourth. Each unlikely event led me to my next. Today, I am convinced that my journey has been steered by forces far greater than my own; in particular, by the support of some remarkable people.

First of all, I would like to thank my thesis supervisors, Professor Randall Davis and Professor Andrew W. Lo. I am incredibly fortunate to have had the privilege of working under both of your esteemed guidance. You taught me what it means to pursue excellence in research and were a constant source of clarity for me. It was a real treat to work with you on the explainability of artificial intelligence, a pressing need for a pivotal field of our time. Most importantly, thank you for your precious time.

I am extremely grateful to the Laboratory for Financial Engineering (LFE) for funding my thesis and supporting me with a research assistantship. This thesis would not have been possible without it. I am also thankful to FICO for providing me the data that served as the foundational bedrock for my work. Arash Nourian and Sue Hubbard at FICO were both invaluable resources who introduced me to the world of credit risk modeling and clarified my every question along the way.

I would also like to thank fate for the relentless twists and turns in the past year. It was my current college-roommate Sam Huang who first introduced me to Professor Lo and the LFE. A few months in, our team encountered a major hurdle when we realized that the data for our research would not arrive on time. It was during this period of extreme uncertainty that Shreyash Agarwal, my freshman-year roommate stumbled across a news article about the Explainable Machine Learning Challenge, a partnership between MIT and FICO. A chain of emails led me to a meeting with Professor Davis, the MIT sponsor for the competition. To my amazing fortune, Professor Davis and Professor Lo found a strong alignment in their research goals and both graciously agreed to supervise my thesis. I received a data set from FICO within a matter of days.

Over the years, I have found the greatest sources of joy and laughter in my parents Rajesh Ram Mishra and Sujata Mishra, my sister Prapti Mishra, my dog Frodo, and my childhood friends Arjun Gandhi and Manu Hegdekatte. In closing, I would like to thank each of them for always making me feel like I'm never too far away from home.

Contents

1 Introduction **21**

1.1 Black-Box Nature Of Deep Learning 21

1.2 Research Objectives 22

1.3 Structure of Thesis and Summary of Key Results 24

2 Overview of Consumer Credit Risk Modeling **35**

2.1 Introduction 36

2.2 Traditional Credit Scoring 36

2.3 Deep Learning In Credit Scoring 37

3 Literature Review **39**

3.1 Motivations for Interpretability 39

3.2 Notions of Interpretability 40

3.2.1 Local Interpretability 40

4 Data **49**

4.1 Description of HELOC Dataset 49

4.1.1 Summary and Construction of the Dataset 49

4.1.2 Glossary of Relevant Credit Modeling Terms 51

4.1.3 Explanation of Predictor Features 51

4.1.4 Permissible Values of Categorical Features 53

4.2 Special Values 54

4.2.1 Dropping Special Values 54

4.2.2 Replacing Special Values with Feature Means 56

4.2.3 Binning Features 56

4.3 Exploratory Analysis 60

4.3.1 Correlation of Features 60

4.3.2 Principal Component Analysis 61

5 Machine Learning Algorithms **65**

5.1 Performance Summary 66

5.2 Model 67

5.2.1 Logistic Regression 67

5.2.2 Random Forest 69

5.2.3 Multi-Layer Perceptron 73

6 Evaluating LIME, a Local Model-Approximation Technique **79**

6.1 Motivation for Approximating a Model Using Piece-Wise Linear Local Approximations 79

6.2 Local Interpretable Model-Agnostic Explanations (LIME) 81

6.2.1 Original Implementation of LIME 83

6.2.2 Shortcomings of LIME 85

6.2.3 Modified Implementation of LIME 86

6.2.4 Evaluating the Original and Modified Implementation of LIME 87

7 Generating Explanations: Rationales and Effective Modifications 97

7.1 Methodology 97

7.2 Examples 98

7.2.1 Example 1: Actionable Steps For A Non-Creditworthy Applicant 98

7.2.2 Example 2: Cautionary Advice For A Creditworthy Applicant 103

7.2.3 Example 3: Actionable Steps For A Non-Creditworthy Applicant, A Problematic Case 106

7.3 Explanatory Utility 112

8 Conclusion and Next Steps 117

8.1 Key Results 117

8.2 Further Work 120

9 Appendix 123

9.1 Calculations for Example 2 123

9.1.1 Average Age of All Lines Of Credit 123

9.2 Calculations for Example 3 124

9.2.1 Average Age of All Lines Of Credit 124

9.2.2 Percentage Of Lines Of Credit Never Delinquent 125

9.2.3 Percentage Of Installment Lines Of Credit 126

9.2.4 Percentage Of Lines Of Credit With Balance 127

List of Figures

1-1 A LIME Linear Approximation of an MLP at a WoE-Encoded Sample Credit Applicant. Each bar is a coefficient in the linear regression. A unit change in the value of a feature with a bar in red and slanted lines will increase the probability of default. A similar change in the value of a feature with a bar in dotted green will decrease that probability. 27

1-2 Rationales for the Prediction of $\Pr(Y = 1|X) = 0.805$ Produced by the LIME Linear Approximation. The features with bars in purple and slanted lines increased the probability of default and those with bars in dotted yellow decreased that probability. 30

1-3 Flow Chart for Generating Domain-Informed Modifications. Once it is determined that a domain-agnostic modification is effective, the explanatory utility of features are considered to produce a domain-informed modification. If found effective as well, the modification is evaluated by human judgment for its pragmatic utility. 33

3-1 Illustration of a LIME Local Approximation [9]. The dashed line is an approximation of the decision surface at the highlighted red point. . . 42

3-2 The Explanation Vector [12] for a Model at a Point. The explanation vector is a gradient that identifies the direction that will produce the greatest change in the label of that point. 44

3-3 Integrated Gradients vs Regular Gradients of Image Labels With Respect To Input Pixels. The Integrated Gradients are able to delineate the pixels that best characterize the camera and the mosque with better sharpness than the traditional gradients. 45

3-4 Example Correction Generated by Zhang et al. [14]. The red plus denotes an adversely classified point. The blue triangular region represents a minimal, stable, and symbolic correction for that data point. The region inside this triangle is classified favorably. 46

3-5 Illustration of Results Produced by Influence Functions [17]. The pictures in the 4x4 grid represent training examples that contributed most to the classification of the gold fish in the test example. 47

3-6 Illustration of Enforcing a Monotonic Constraint Between User Happiness and Distance to Cafe [19]. The purple line represents the decision boundary learned from this data. The green line represents the decision boundary learned with the monotonic constraint that User happiness must non-increasing with increasing Distance To Cafe. 48

4-1 Sampling Window of the HELOC Dataset. Applicants were able to use their HELOC for anywhere between one and three years before their performance was evaluated. 50

4-2 Distribution of Risk Classification. The number of Creditworthy and Non-Creditworthy applicants is roughly evenly-split by construction in this dataset. 50

4-3 Venn Diagram of Records containing at least one special value. $7,957$ applicants contain at least one special value. Hence, it is infeasible to drop them all. 55

4-4 Illustration Of One Hot Encoding [20]. A feature with four possible values is mapped into a four dimensional space where exactly one dimension per vector is "hot" i.e., has a unit value. 58

4-5 Histogram of Feature - Months since the very first line of credit was established (MSinceFirstLOC). The feature is approximately normally distributed and we see a large concentration of values at 0. 59

4-6 Countplot of WoE Encoded Feature - Months since the very first line of credit was established (MSinceFirstLOC). The continuous values of the feature in Figure 4-5 are discretized into WoE values. 60

4-7 23 x 23 Correlation Matrix. The lighter shades represents highly correlated feature pairs. As expected, the diagonal has the lightest shade because it represents the correlation of a feature with itself. 61

4-8 3D Principal Components Analysis (1 of 3). The green and red dots are not readily separable. 62

4-9 3D Principal Components Analysis (2 of 3). The green and red dots are not easily separable. 63

4-10 3D Principal Components Analysis (3 of 3). The green and red dots are not easily separable. 63

4-11 Explained Variance vs Number of Principal Components. We see that 15 principal components are sufficient to represent 95% of the variance of the 23-dimensional dataset. 64

5-1 Comparison of Decision Boundary Between Logistic Regression and Linear Regression. Unlike linear regression, the output of logistic regression is constrained to be between 0 and 1. 68

5-2 Relative Feature Importance Computed from Logistic Regression. The regression is trained on scaled Truncated Data so that coefficient values can be directly compared. We see that the number of requests for new lines of credit in the past 6 months is identified as the most significant feature. 70

5-3 Illustration of Random Forest. To classify a new data point, the Random Forest aggregates the predictions from each of its constituent decision trees and outputs the majority vote as its classification. In this example, the point receives a majority classification of Class C. 72

5-4 Accuracy of Random Forest vs Number of Estimators on WoE Data. The maximum accuracy is achieved once there are at least 30 decision trees. 73

5-5 Relative Global Feature Importance for Random Forest. This ordering is determined by measuring the average decrease in Gini impurity produced by each feature across all decision trees. 74

5-6 A Multi-Layer Perceptron [22]. It has an input layer with three neurons, one hidden layer with four neurons, and an output layer with one neuron. 75

5-7 ReLU(z) = max(0,z) [23]. The function maps any positive integer to itself and any negative integer to zero. 76

6-1 A LIME Linear Approximation of an MLP at a WoE-encoded Sample Credit Applicant. Each bar is a coefficient in the linear regression. A unit change in the value of a feature with a bar in red and slanted lines will increase the probability of default. A similar change in the value of a feature with a bar in dotted green will decrease that probability. 82

6-2 Illustration of LIME Algorithm In 2 Dimensions For Simplicity. . . . 84

6-3 Flow Chart for Generating Effective Modifications. Once a modification is generated, we check if it is effective. If yes, we terminate with a solution. Otherwise, we either increase or decrease the value of p^{target} based on the desired label. If p^{target} reaches its limits of 0 or 1 and no effective modification has been found, we terminate with no solution. 91

7-1 Rationales for the Prediction of $\Pr(Y = 1|X^{(1)}) = 0.805$ Produced by the LIME Linear Approximation. The features with bars in purple and slanted lines increased the probability of default and those with bars in dotted yellow decreased that probability. 100

7-2 Rationales for the Prediction of $\Pr(Y = 1|X^{(2)}) = 0.17$ Produced by the LIME Linear Approximation. The features with bars in purple and slanted lines increased the probability of default and those with bars in dotted yellow decreased that probability. 105

7-3 Rationales for the Prediction of $\Pr(Y = 1|X^{(3)}) = 0.695$ Produced by the LIME Linear Approximation. The features with bars in purple and slanted lines increased the probability of default and those with bars in dotted yellow decreased that probability. 108

7-4 Flow Chart for Generating Domain-Informed Modifications. Once it is determined that a domain-agnostic modification is effective, the explanatory utility of features are considered to produce a domain-informed modification. If found effective as well, the modification is evaluated by human judgment for its pragmatic utility. 115

List of Tables

1.1 5-Fold Cross-Validation Performance of Machine Learning Models Trained On Truncated and WoE Data. The MLP trained on WoE data has the best mean test accuracy of 74.7%. 26

1.2 Comparison of Modification Power Between Original and Modified LIME. The modified implementation does significantly better when p^{target} is close to the decision boundary. As p^{target} decreases, the difference between the Modification Power of both implementations is not statistically significant. 29

4.1 Explanation of Categorical Features Values for MaxDelqLast12M and MaxDelqEver. Subsets of these values have particular orderings (e.g. C-F) but there is no strict ordering over the entire set. 53

4.2 Distribution of Special Values Across Columns. These special values are concentrated among 9 of our 23 input features. 56

4.3 Illustration of Binning a Sample Feature. Each feature value maps uniquely to a single bin. 57

4.4 Weight of Evidence Encoding For A Sample Feature. WoE values are discrete and no two bins share the same WoE value. 59

5.1 5-Fold Cross-Validation Performance of Machine Learning Models Trained On Truncated and WoE Data. The MLP trained on WoE data had the best mean test accuracy of 74.7%. 67

5.2 5-Fold Cross-Validation Performance of MLP Architectures on WoE Data. The architecture with the best mean test performance was (3,2,2). We see that the performance of MLP does not improve with deeper architectures beyond a threshold, suggesting that the HELOC dataset may not be suitable for training a large number of parameters. 77

6.1 Comparison of Constraint Violations Among 5,000 Perturbed Points Sampled from the Original and Modified Implementations of LIME. The points sampled from the modified implementation of LIME have fewer violations in 7 out of the 12 constraints. 89

6.2 Count of Constraint Violations by Perturbed Data Generated from Original and Modified Implementations of LIME. The average violations per perturbed point is lower for the modified implementation than that of the original implementation. 90

6.3 Comparison of Modification Power Between Original and Modified LIME. The modified implementation does significantly better when p^{target} is close to the decision boundary. As p^{target} decreases, the difference between the Modification Power of both implementations is not statistically significant. 94

7.1 Original and WoE Feature Values of The Non-creditworthy Applicant $X^{(1)}$. $MaxDelqEver = E$ means the credit applicant has been 60+ days delinquent ever. $MaxDelqLast12M = G$ means their maximum delinquency in the last 12 months is is unknown. 99

7.2 Original and WoE Feature Values of The Creditworthy Applicant $X^{(2)}$. The values of MaxDelqEver and MaxDelqLast12M mean that the applicant has never had a delinquency. 103

7.3 Original and WoE Value Of The Non-Creditworthy Applicant. $MaxDelqEver = G$ and $MaxDelqLast12M = G$ both means it is not known if the applicant has ever had a delinquency in the respective time lines. 106

Chapter 1

Introduction

This chapter serves to introduce the research objectives of this thesis and to summarize its key findings. We begin by describing the black-box nature of deep learning. Then, we explore the need for interpreting the prediction of deep learning models in a specific domain, namely credit risk modeling. To do this, we formalize the explanations that would be most useful for a credit applicant.

Every applicant could usefully understand why they were classified the way they were. An adversely classified applicant would also be interested in actionable advice to reach a favorable classification and a favorably classified applicant in guidance on how not to be turned down the next time. This context provides a useful grounding for generating such explanations using model-approximation techniques. Through our results, we show how numerical techniques can be combined with domain-knowledge to generate effective and pragmatically-useful modifications.

1.1 Black-Box Nature Of Deep Learning

Recent advances in computing, algorithmic innovations, and an explosion of data have contributed to the growing success of deep learning. This sub-field of machine learning is concerned with training large neural networks with many hidden layers. Unlike traditional machine learning techniques like logistic regression and support vector machines, deep learning models can automatically learn non-linear representations and interactions of input features from large datasets.

In 2012, Krizhevsky et al. [1] demonstrated state-of-the-art performance in scene recognition (15.4% top-5 error rate) by using a deep neural network that consisted of 60 million parameters and was trained on 1.2 million images. This breakthrough inspired researchers to experiment even further with deeper architectures. In 2015, Simonyan [2] achieved a new state-of-the-art (6.8% top-5 error rate) by training an ensemble of two deep neural networks with 144 million parameters each. Although the large number of parameters of such deep neural networks allow them to learn complex relationships and achieve superior performance, they are also responsible for their black-box decision-making. A human simply cannot comprehend the meaning of 60 million parameters without any intuitive interpretations, let alone an ensemble of 144 million parameters.

1.2 Research Objectives

Before attempting to explain the prediction of a machine learning model, it is important to answer who the explanation is for and what purpose it aims to serve. In this thesis, we focus on the credit risk modeling domain, a real-world example of an important classification task. Specifically, we generate explanations most useful for a credit applicant. Therefore, of the many possible goals for an explanation, we choose two. These goals are pertinent to understanding the prediction of any machine learning model.

1. Justifications for the classification chosen.

2. Advice of two sorts:

(a) How to change the classification (e.g. if denied, how a credit applicant can become Creditworthy).

(b) How to maintain the classification (e.g. if deemed Creditworthy, what precautions should a credit applicant take to not be turned down in the future).

Let us denote our task more formally. Assume we have a machine learning model f that takes as input an n dimensional vector X and predicts a binary label Y.

$$f(X) = \begin{cases} 1, & \text{if } \Pr(Y = 1|X) > 0.5 \\ 0, & \text{otherwise} \end{cases}$$

To generate our explanations, we are interested in two tasks.

1. **Rationales** - We want to identify the top k features that contributed most to a classification, where k is a parameter selectable by a user.

   We can compute this ordering by using a contribution attribution function A that takes as input a model f and a particular input vector X and outputs an n dimensional vector C, where C_i represents the relative contribution of feature i for the classification of X.

$$A(f, X) = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}$$

   We can use this contribution vector C to identify the k features with the largest contributions by face value or absolute value.

2. **Effective Modifications** - We want to specify a set of minimal changes to the feature values of a sample point such that its classification reverses.

Given f and X, we must solve for $X^{modified}$ in the optimization in Equation 1.1.

$$\begin{aligned} \text{minimize} \quad & \left\|X^{modified} - X\right\| \\ \text{subject to} \quad & f(X) \neq f(X^{modified}) \end{aligned} \tag{1.1}$$

Then,

$$\delta = X^{modified} - X$$

gives us the set of minimal changes that would reverse the classification of X.

While we illustrate these objectives with respect to credit risk modeling, they generalize more broadly. Regardless of the domain, one might be interested in attributing the prediction of a machine learning model among its input features and generating actionable steps to reverse the classification of a sample point.

1.3 Structure of Thesis and Summary of Key Results

In this section, we provide the chapter-wise structure of the thesis and summarize its key results.

Chapter 2: Overview of Consumer Credit Risk Modeling

We begin with an overview of credit risk modeling. We explain why logistic regression is commonly used for the binary classification task and detail the potential for deep learning. We also outline the explainability regulations for algorithmic decision-makers and the resulting compliance challenges.

Chapter 3: Literature Review

We survey the existing literature to evaluate previous attempts to interpret the predictions of machine learning models. We also discuss our choice of using LIME, a model-approximation technique to generate explanations.

Chapter 4: Data

In this thesis, we argue that domain-knowledge is necessary to generate pragmatically-useful explanations. This chapter thus provides an important context for evaluating the pragmatic utility of explanations generated subsequently. We introduce the Home Equity Line Of Credit (HELOC) dataset used in our analyses and describe its 23 features in detail. We also describe how we handle the special values in the dataset using two approaches. One, we drop all features containing at least one special value. We refer to this version of the dataset as Truncated data. Two, we re-code all features using a technique called Weight Of Evidence (WoE) encoding. We refer to this version as WoE data.

Chapter 5: Machine Learning Algorithms

This chapter benchmarks the performance of three machine learning algorithms trained on the Truncated and WoE versions of the HELOC dataset: Logistic Regression (LR), Random Forest (RF), and Multi-Layer Perceptron (MLP). Table 1.1 shows that the MLP trained on WoE data is the best-performing model with 74.7% test accuracy. We also see that all three models have a superior test performance on WoE data versus Truncated data. We choose the MLP as the foundational test-bed for our interpretability work because of its superior performance and our interest in explaining the predictions of deep learning models.

Dataset	Fold	LR		RF		MLP	
		Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
Truncated	Train	0.724	0.002	0.99	0	0.666	0.004
Truncated	Test	0.723	0.008	0.706	0.007	0.664	0.005
WoE	Train	0.746	0.002	0.99	0	0.749	0.001
WoE	Test	0.744	0.010	0.723	0.007	0.747	0.008

Table 1.1: 5-Fold Cross-Validation Performance of Machine Learning Models Trained On Truncated and WoE Data. The MLP trained on WoE data has the best mean test accuracy of 74.7%.

Chapter 6: Evaluating LIME, a Local Model-Approximation Technique

We consider simplicity to be a key element of interpretability. In this chapter, we claim that linear regression models are simple because of their decomposability. To interpret the predictions of a black-box MLP model, we choose to approximate it with several piece-wise linear local approximations.

Local Interpretable Model-Agnostic Explanations (LIME) is a model-agnostic technique that can approximate the decision boundary of a machine learning model by constructing a locally-weighted linear regression in the neighborhood of a particular data point. Figure 1-1 visualizes the regression coefficients computed by LIME to approximate the MLP at a sample point in the WoE space.

Each coefficient in Figure 1-1 represents the change in the probability of default of the applicant i.e., $\Pr(Y = 1|X)$ produced by a unit change in the WoE value of the corresponding feature, holding other feature values constant. A unit change in the WoE value of a feature with a coefficient in red and slanted lines increases the probability of default of the applicant. A similar change for a feature with a coefficient in dotted green decreases the probability of default of the applicant. It is

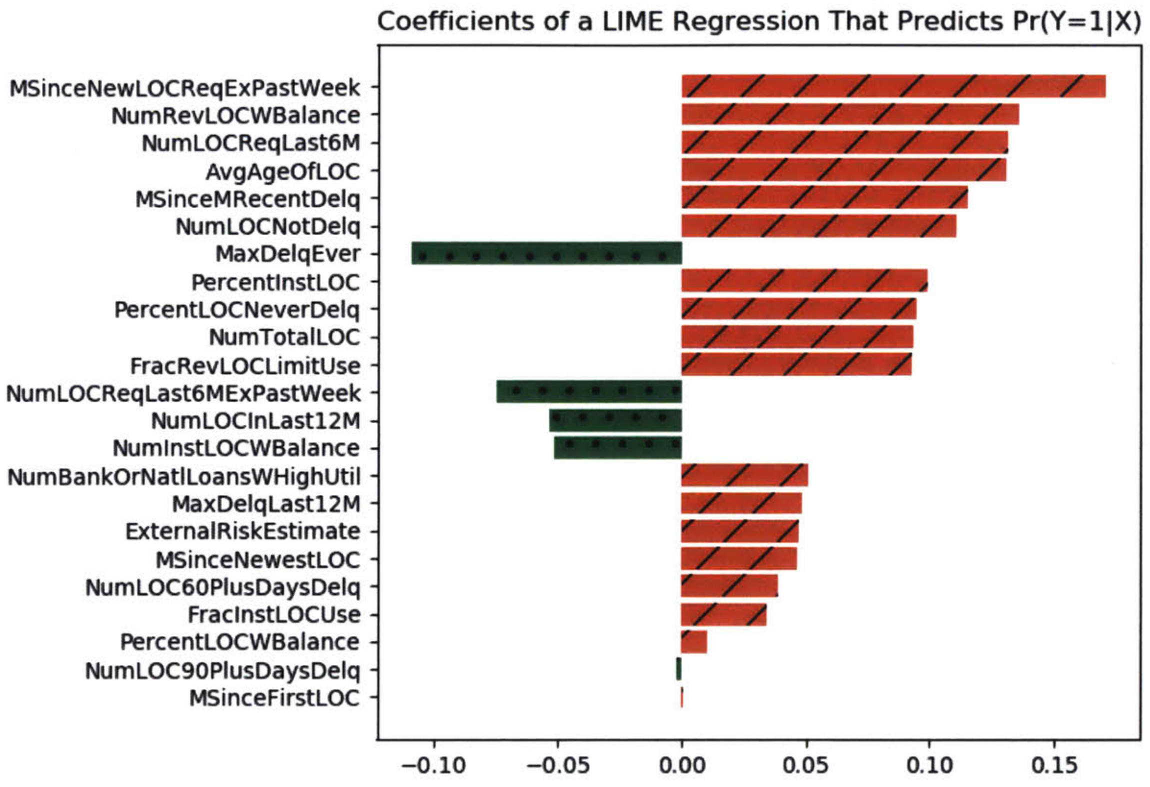

Figure 1-1: A LIME Linear Approximation of an MLP at a WoE-Encoded Sample Credit Applicant. Each bar is a coefficient in the linear regression. A unit change in the value of a feature with a bar in red and slanted lines will increase the probability of default. A similar change in the value of a feature with a bar in dotted green will decrease that probability.

worth clarifying that these coefficients in WoE space should not be used directly to infer the contribution of features in the original space for this particular prediction. Section 6.2 describes the correct way to interpret the coefficients.

LIME generates a neighborhood of points by perturbing the data point of interest. The original implementation of LIME assumes independence of features while sampling these points. We implement an alternate version that relaxes this assumption. Our hypothesis was that an implementation based on perturbed points with greater validity will also produce a better approximation. We observe that the modified implementation outperforms the original implementation in certain scenarios, as measured by two criteria - Fidelity and Modification Power. While this is not true in all scenarios, we believe that accounting for the correlation of features in LIME's algorithm, as we did, is the correct approach when dealing with datasets with highly correlated features.

We refer to the Fidelity of an approximation as a measure of how often the approximation matches the prediction of the underlying model at the data point of interest. The approximations generated by the modified implementation achieve a slightly higher Fidelity i.e. $\frac{1937}{1973} = 0.981$ vs the $\frac{1934}{1973} = 0.980$ for those generated by the original implementation. This small difference is not statistically significant.

We use the term Modification Power to mean the rate with which we are able to reverse the classifications of a set of points by changing their feature values using modifications derived from linear approximations. We implement a generate and test algorithm to suggest effective modifications intended to reverse the classification of a given input point.

The classification of a credit applicant depends on their computed probability of default with a threshold of 0.5. If the computed probability is greater than 0.5, the applicant is deemed Non-creditworthy and turned down. With each iteration of our modification algorithm, we decrease (increase) the target probability of default for that applicant. A smaller value is chosen for p^{target} if the desired label is favorable and a larger value if the desired label is adverse. The algorithm then identifies a set of changes that are likely to alter the probability of default of the modified data point to p^{target}. Then, the algorithm determines whether this goal was indeed accomplished. If so, the algorithm terminates. Otherwise, it chooses a more extreme value for p^{target} and repeats. It fails when no effective modifications are found once p^{target} has reached its limit of 0 or 1.

Table 1.2 shows the effect of varying p^{target} on the rate of successfully changing the classification of data points from adverse to favorable. Choosing a smaller p^{target} means we are looking to push a data point further into the favorable region, farther from the decision boundary. As expected, the success of reversing the classification of a set of adversely classified points increases as we decrease p^{target}. Another interesting result is that the modified implementation of LIME has a significantly better Modification Power than that of the original implementation when p^{target} is close to the edge of the decision boundary at 0.5. The significance of this difference decreases as we decrease p^{target}. This makes sense intuitively because coarse approximations may be sufficient to guide a data point deep across the other side of the decision boundary.

p^{target}	Original LIME Modification Power (1)	Modified LIME Modification Power (2)	Difference (2) - (1)
0.5	$\frac{589}{782} = 0.753$	$\frac{636}{782} = 0.813$	$\frac{47}{782} = 0.060$
0.49	$\frac{682}{782} = 0.872$	$\frac{710}{782} = 0.908$	$\frac{28}{782} = 0.036$
0.48	$\frac{747}{782} = 0.955$	$\frac{755}{782} = 0.965$	$\frac{8}{782} = 0.010$
0.47	$\frac{766}{782} = 0.980$	$\frac{768}{782} = 0.982$	$\frac{2}{782} = 0.002$
0.46	$\frac{777}{782} = 0.993$	$\frac{774}{782} = 0.989$	$\frac{-3}{782} = -0.003$
0.45	$\frac{780}{782} = 0.997$	$\frac{777}{782} = 0.993$	$\frac{-3}{782} = -0.003$
0.44	$\frac{782}{782} = 1.000$	$\frac{781}{782} = 0.998$	$\frac{-1}{782} = -0.002$

Table 1.2: Comparison of Modification Power Between Original and Modified LIME. The modified implementation does significantly better when p^{target} is close to the decision boundary. As p^{target} decreases, the difference between the Modification Power of both implementations is not statistically significant.

Chapter 7: Generating Explanations: Rationales and Effective Modifications

We generate the two desired explanations, namely rationales and effective modifications, for three sample credit applicants. Using our algorithm, we show that domain-knowledge must be used to produce modifications of pragmatic utility.

Figure 7-1 visualizes the rationales for a credit applicant who was deemed Non-creditworthy with $\Pr(Y = 1|X) = 0.805$. The features with bars in purple and slanted lines increase the probability of the applicant being classified as Non-creditworthy while the features with bars in dotted yellow decrease that probability.

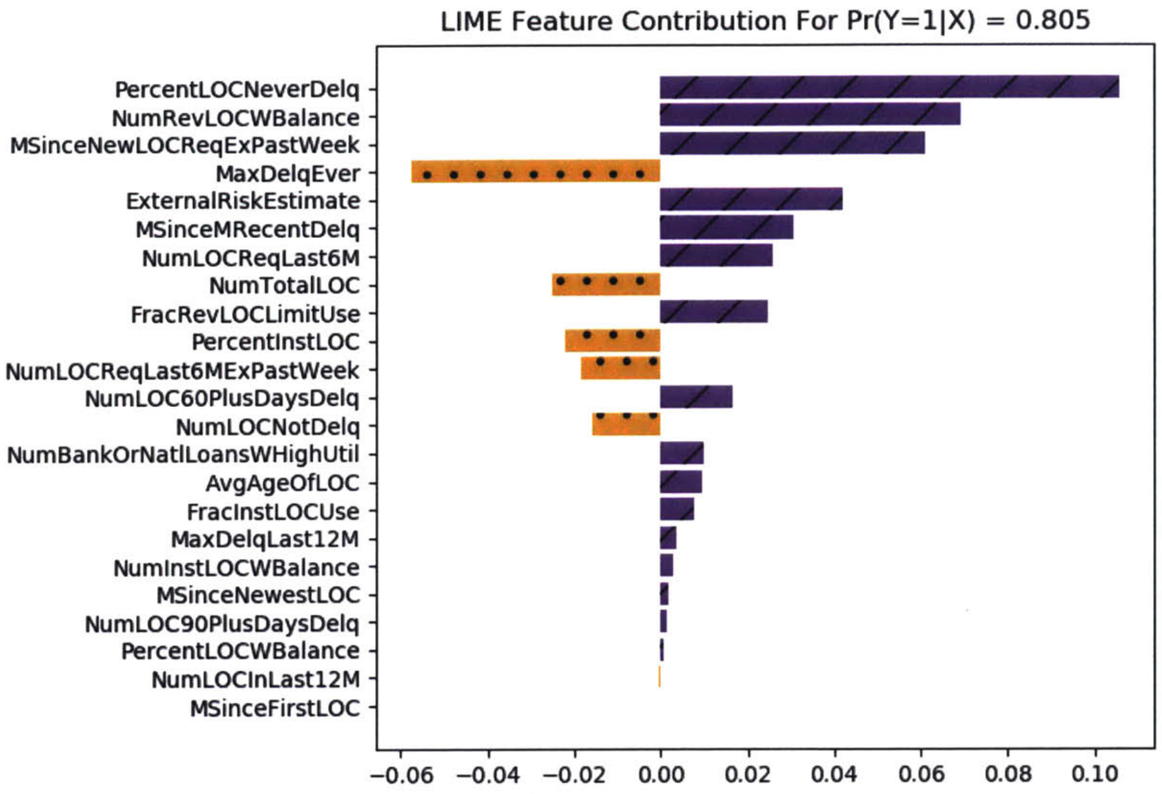

Figure 1-2: Rationales for the Prediction of $\Pr(Y = 1|X) = 0.805$ Produced by the LIME Linear Approximation. The features with bars in purple and slanted lines increased the probability of default and those with bars in dotted yellow decreased that probability.

As per the requirements of The Fair Credit Reporting Act, we can generate the four key factors that adversely affected the rejected credit applicant. The rationales are:

1. Too many of their lines of credit have been delinquent at least once (PercentLOCNeverDelq).

2. They have too many revolving lines of credit with outstanding balances (NumRevLOCWBalance).

3. They have a poor global credit score (ExternalRiskEstimate).

4. They requested for a new line of credit too recently (MSinceNewLOCReqExPastWeek).

We show that the modifications suggested by our algorithm, though effective, are not necessarily of pragmatic utility. They could have three possible types of shortcomings.

1. They suggest changes to the value of a feature that is in fact not controllable by a credit applicant.
2. They suggest practically infeasible changes to the value of a feature, even if it is controllable.
3. They do not account for the second-order effects of changing feature values, resulting from interdependencies among features.

These shortcomings cannot be dealt with by solely relying on numerical techniques agnostic to the domain of the data. Instead, we must use domain-knowledge and consider the *explanatory utility* of each feature in an explanation. To generate pragmatically-useful modifications, we augment our algorithm with domain-knowledge in three ways:

1. **Account for the Controllability of Features** - We explicitly categorize each predictor feature as controllable (e.g. the number of total lines of credit) or non-controllable (e.g. the external risk estimate). We constrain our algorithm to suggest changes only to the values of features that are controllable.
2. **Account for Second-Order Effects** - We consider the interdependencies between features and account for them programmatically. For instance, if a new line of credit is opened, the average age of all lines of credit will change in a computationally predictable manner, among other effects.

3. **Evaluate the Feasibility of Changing Feature Values** - We use human judgment to determine if an effective and domain-informed modification is also pragmatically-useful.

These changes are reflected in the flowchart presented in Figure 1-3. Once a *domain-agnostic modification* is generated, we determine whether it is effective. If yes, we use domain-knowledge to constrain the modification to changes in controllable features only and account for second-order effects due to interdependencies among features. We then produce a *domain-informed modification.* We test for its effectiveness programmatically and finally, use human judgment to evaluate its pragmatic utility.

Chapter 8: Conclusion and Next Steps

We conclude with a summary of the key findings of the thesis and suggest areas for further research. These include evaluating other model-approximation techniques such as SHAP, furthering our modified implementation of LIME, and extending the capabilities of our modification algorithm.

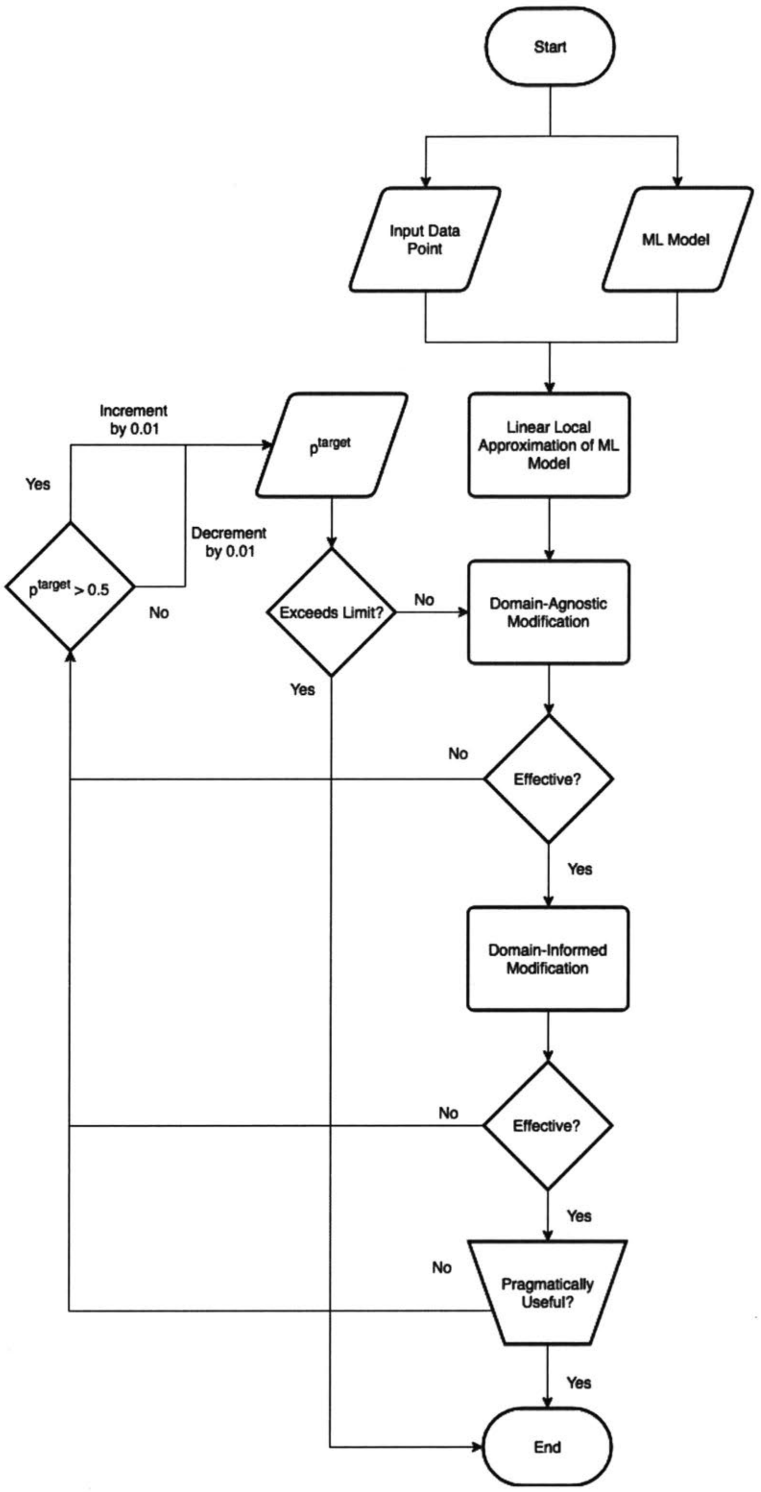

Figure 1-3: Flow Chart for Generating Domain-Informed Modifications. Once it is determined that a domain-agnostic modification is effective, the explanatory utility of features are considered to produce a domain-informed modification. If found effective as well, the modification is evaluated by human judgment for its pragmatic utility.

Chapter 2

Overview of Consumer Credit Risk Modeling

The consumer credit risk modeling industry has historically relied on logistic regression to predict the probability of default of a credit applicant. The technique remains popular because it delivers strong accuracy through a simple and transparent approach. However, recent advances in deep learning present an opportunity to achieve unprecedented accuracy in credit risk modeling at the cost of interpretability. While there is a strong case for prioritizing performance, this chapter serves to highlight that the trade-off between performance and interpretability cannot be made freely. Industry regulations impose strict guidelines on algorithmic decision-makers, especially when they classify credit applicants adversely. This challenge motivates the need for interpreting the decision-making of black-box machine learning models in this particular domain.

2.1 Introduction

Consumer credit ratings play an outsized role in the world today. Lenders use them to decide whether to grant credit, to structure the terms of credit, and to offer further financial products. This applies across a wide spectrum of credit offerings such as mortgages, car loans, credit cards, and home equity lines of credit. Lenders use credit ratings extensively because they are accurate, cheap, and quick proxies for the creditworthiness of individuals, thereby enabling them to manage their risk efficiently.

As of March 31, 2018, the total household indebtedness in U.S.A was at a peak of \$13.21 trillion [3]. If credit rating models misclassify defaulters as non-defaulters (Type I error), lenders could be exposed to enormous losses. On the other hand, if the models misclassify non-defaulters as defaulters (Type II error), lenders lose business. Given the massive amount of outstanding household debt in America, even a small percentage improvement in the accuracy of credit ratings can have a sizeable financial impact.

2.2 Traditional Credit Scoring

The fundamental goal of credit scoring is to determine whether a credit applicant will default on their loans. Simply put, it is a binary classification task that labels credit applicants as Creditworthy or Non-creditworthy. A Creditworthy applicant is likely to repay their financial obligation while a Non-creditworthy applicant is not.

Historically, credit ratings have been computed using logistic regression. Attributes of a credit applicant are first carefully chosen as features using domain knowledge and feature selection techniques. The goal is to identify the ones that are most predictive of future default. The feature values of each applicant are then transformed to the same scale using techniques like Weight Of Evidence (WoE) encoding. This standardization allows for easy comparison of the importance of features using the regression coefficients. Finally, the transformed features are fed into a logistic regression model that outputs the applicant's probability of default. If this probability falls below a threshold specified by the lender, the applicant is classified as Creditworthy and otherwise as Non-creditworthy.

Using logistic regression to solve the binary classification task offers several benefits: results backed by a well-understood theory, simplicity, and powerful modeling capabilities. As described in Section 5.2.1, logistic regression is computed using Maximum Likelihood Estimation, a technique with a well-founded mathematical underpinning. Logistic regression model is simple because of its decomposability into a set of regression coefficients. A linear combination of these regression coefficients sum to the log odds of $\Pr(Y = 1|X)$, where Y is the binary outcome variable and X is the input variable. The equation below be used to determine the contribution of each feature for a particular prediction:

$$\log[\frac{Pr(Y = 1|X)}{1 - Pr(Y = 1|X)}] = \beta_0 + \beta_1 \cdot X_1 + \beta_2 \cdot X_2 + ... + \beta_k \cdot X_k$$

Lastly, logistic regression can be used with feature engineering to model the relationship between non-linear transformations of inputs and the output probability. However, it takes expert knowledge to select relevant features and to identify appropriate non-linear transformations. This process can be tedious, time-intensive, and prone to errors in human judgment.

2.3 Deep Learning In Credit Scoring

Machine learning is the science of getting computers to act without being explicitly programmed [4]. In recent years, the sub-field of deep learning has garnered significant attention for achieving unprecedented performance in classification tasks such as object and speech recognition. An appealing aspect of deep learning is that it requires little manual feature engineering. In fact, the very reason these models perform so well is that they learn feature representations and non-linear interactions from large datasets instead of relying on expert human judgment. By learning directly from large amounts of data, these models can classify data points using complex decision boundaries and achieve superior accuracy.

A function is a relation from a set of inputs to a set of possible outputs where each input is related to exactly one output [5]. We can view a credit risk model as a function that uniquely determines the relation between the attributes of a credit applicant and the probability that they are Non-creditworthy. A variety of machine learning techniques can be used for credit risk prediction because they are all different approaches to the same goal of determining an accurate mapping from a set of input variables to a set of target outputs.

While researchers have demonstrated the promising potential of deep learning to predict consumer credit risk [6], the credit risk modeling industry has been slow to adopt the technique due to its black-box nature. The primary obstacle is regulatory compliance. The Fair Credit Reporting Act (1970) mandates that lenders must be able to disclose up to four key factors that adversely affected the credit score of a rejected consumer. More recently, the European Union's General Data Protect Regulation (2018) created a *right to explanation*, whereby a user can ask for an explanation of an algorithmic decision that was made about them [7].

To comply with these regulations, a credit risk model should be able to attribute its decision among contributing factors. As outlined above, logistic regression has this property because of its decomposability. Each regression coefficient corresponds to a feature and can be used to determine that feature's contribution towards a prediction. However, the potentially vast parameters of deep learning models do not have intuitive interpretations and cannot be used directly to compute feature contributions. This creates an opacity in the decision-making of deep learning models that is in conflict with the transparency required by regulations in the credit risk modeling industry.

Chapter 3

Literature Review

This chapter explores the various motivations and notions of interpretability. We describe previous attempts to make machine learning models more understandable and provide a point of contrast for our research.

3.1 Motivations for Interpretability

The call for interpretability in machine learning (ML) is clear. Doctors who diagnose patients using ML would want to verify that spurious correlations are not being used to make causal inferences. Regulators would want to ensure that ML used to grant credit does not discriminate by race or gender. Engineers building a self-driving car would want to understand the vulnerabilities of its perception ML to adversarial attacks. Evidently, a black-box ML model that simply generates predictions cannot directly satisfy these needs. We require insights into its decision-making. Although these scenarios make a seemingly united argument for the need for explainable ML systems, they are motivated by the need to confirm distinct properties, namely causality, fairness, and robustness [8]. These distinct needs entail distinct notions of explainability and thus potentially require different interpretability techniques.

As introduced in Chapter 1, we motivate our objectives of producing feature attribution and modifications through the domain of credit risk modeling. In particular, we are concerned with generating explanations most useful for a credit applicant. We refer to feature attribution as the task of determining the contribution of input features towards a prediction by an ML model. We define modifications as a set of minimal changes to feature values of a sample point to change its classification to the desired label. We explore previous work on interpretability that either directly accomplished these objectives or could be adapted to do so.

3.2 Notions of Interpretability

There are two broad notions of interpretability - Global and Local. Global interpretability means understanding how a model works. For instance, monotonicity provides global interpretability because it is an invariant on the relationship between input features and the target predicted value over the entire observation set. In contrast, local interpretability is a set of rationales for a specific decision. Our objectives of producing feature attributions and modifications for particular predictions are primarily concerned with local interpretability.

3.2.1 Local Interpretability

Linear approximations of models can be used to produce feature attributions. In a linear regression, feature attribution can be computed by examining its coefficients, weighted by the values of features in a given input. The regression coefficients can be also used to help suggest modifications. Each regression coefficient can be interpreted as the change in the output prediction caused by a unit change in the corresponding feature value, holding other feature values constant. Hence, these coefficients can be used to inform changes to feature values to appropriately alter the output prediction.

LIME and gradients are two approaches to generating linear approximations to an underlying model. One of LIME's limitations is that the regression coefficients of its linear approximation cannot be interpreted reliably in the presence of highly correlated features, as is the case for any linear regression. SHAP is an alternate local interpretability technique that addresses this shortcoming. It can attribute the contribution of features for a particular prediction, robust to the presence of highly correlated features. Alternatively, influence functions, a technique from robust statistics provides a different approach to explaining predictions: by attributing importance among examples in the training set.

In this section, we outline the theory of these local interpretability techniques, illustrate their outputs with examples, and suggest how they could be applied to the research objectives of this thesis. The common thread among these techniques is that they attempt to explain a model's predictions insights using the vocabulary of the features that have been chosen to represent the data point. While it is not necessary that input features always have intuitive meanings, the model creator usually has significant control over their selection.

Linear Approximation of A Model Using LIME

Local Interpretable Model-Agnostic Explanations (LIME) [9] is a local interpretability technique that can approximate a machine learning model at a data point of interest. Its key intuition is that it is much easier to approximate a black-box model using several piece-wise linear local approximations as opposed to a single linear global approximation.

Figure 3-1 illustrates how a LIME approximation is constructed. The lighter and darker regions of the figure correspond to different sides of a decision surface. In this figure, the LIME algorithm constructs a linear approximation of the decision surface in the neighborhood of the highlighted red point. The blue dots and red pluses around that data point represent perturbed points in its vicinity. The points closer to that point are up-weighted than ones further away. A linear regression, represented by the dashed lines, is then trained to best classify these points, given their respective weights. This local weighting ensures that the regression approximates the underlying decision surface best at the input data point. LIME is appealing because it is model-agnostic. It does not require the underlying model to be differentiable or belong to a particular function class. However, it shares the same shortcoming of any linear regression: the assumption that input features are independent. Due to this, the presence of multicollinearity may affect the interpretation of the coefficients of the LIME approximation.

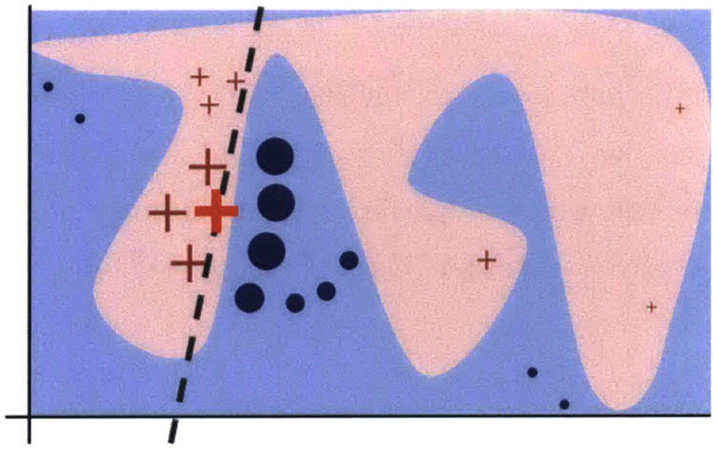

Figure 3-1: Illustration of a LIME Local Approximation [9]. The dashed line is an approximation of the decision surface at the highlighted red point.

Feature Attributing of A Prediction Using SHAP

Unlike LIME, SHAP can compute feature attributions that are robust to highly correlated features. Its drawback, however, is that cannot be interpreted as a linear approximation, which limits its utility in informing modifications. SHAP was inspired by the game theory concept of Shapley Values: If multiple players with differing skills collaborate in a coalition to receive a collective payoff, what is the fairest way to divide up that payoff among the players? The answer is finding each player's marginal contribution, averaged over every possible sequence in which the players could have been added to the group [10]. The same idea can be applied to compute feature contributions for a local prediction. The payout is the prediction by the model and the players are the different features that collaborate to receive that payout. Each Shapley value is, therefore, the average contribution of a feature value towards the prediction in different coalitions. Lundberg and Lee [11] claim that the coefficients of the locally-weighted model generated by LIME are in fact approximations of Shapley Values resulting from the choice of a specific weighting kernel.

Linear Approximation Of A Model Using Gradients

The gradients of a machine learning model can be used to approximate it linearly. A gradient points in the direction of the greatest rate of increase of a function. Therefore, the gradient of the target output label with respect to the input features can inform the direction for greatest change towards that label for a data point. This could be used to compute a modification. In a similar spirit, Baehrens et al. (2010) [12] coined the Explanation Vector, a local gradient that characterizes how a data point has to be moved to change its predicted label. In Figure 3-2, (a) visualizes binary-labeled training data and (b) illustrates the decision surface of a model trained on this data. The white arrows in (c) represent the gradients i.e. the Explanation Vector.

In 2017, Sundararajan et al. [13] developed Integrated Gradients as a superior approach to computing gradients of deep neural networks. They showed that the prediction function of a neural network may flatten in the vicinity of certain inputs, resulting in incorrectly small gradients. They show that gradients computed by Integrated Gradients do a better job in identifying pixels most responsible for a classification than do traditional gradients (See Figure 3-3).

Using gradients and linear programming, Zhang et al. (2018) [14] present an algorithm to provide local explanations using minimal, stable, and symbolic corrections. They sought to provide actionable insights on how to cause a prediction to move from an undesirable class to a desirable class. Their approach showed promising

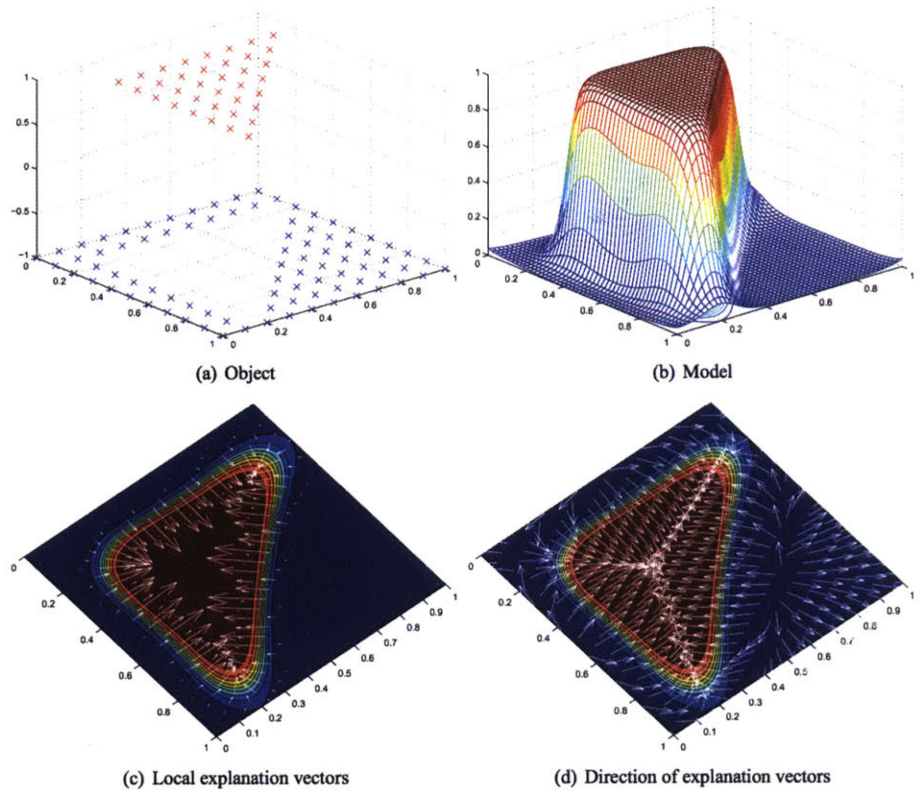

Figure 3-2: The Explanation Vector [12] for a Model at a Point. The explanation vector is a gradient that identifies the direction that will produce the greatest change in the label of that point.

results with proven mathematical guarantees on small neural networks with ReLU activation functions. The limitations of their approach were that they required the underlying model to be differentiable and that they did not account for potential inter-dependencies between input features. As a result, the modifications produced may not be of pragmatic utility. These broader issues are discussed in detail in 7.2.3.

They defined a correction as minimal if it was as similar as possible to the original input. It was symbolic and stable if it was a neighborhood of points for which the outcome was also desirable rather than just a single point. Figure 3-4 illustrates one example of the output of their algorithm. The red cross is an adversely classified point and the blue triangular region represents the stable correction region computed.

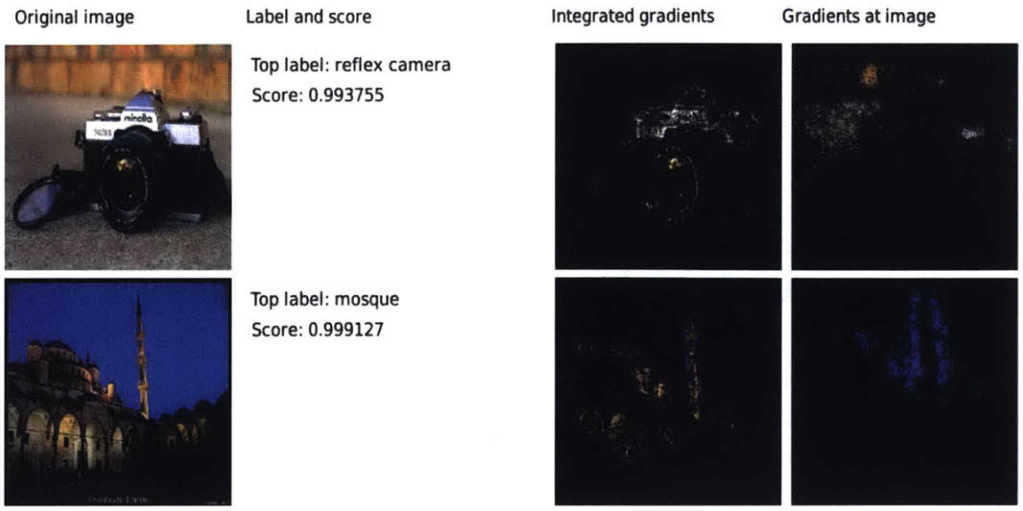

Figure 3-3: Integrated Gradients vs Regular Gradients of Image Labels With Respect To Input Pixels. The Integrated Gradients are able to delineate the pixels that best characterize the camera and the mosque with better sharpness than the traditional gradients.

Explaining with Examples Using Influence Functions

Influence functions is a technique from robust statistics that tells us how a model's parameters change as a training point is up-weighted infinitesimally [15]. Koh et al. [16] applied this technique to trace a black-box model's prediction back to its training data, thereby identifying training points most responsible for a given prediction. In essence, this technique offers local interpretability into a prediction by using examples as explanations. It identifies the training data that contributed the most to the model's prediction. However, such an explanation does not explicitly identify the features most influential to the classification of the sample point and the examples in the training set. The burden instead shifts onto the recipient of the explanation. While a human can visually inspect images produced as explanations, the approach is infeasible with large numerical data. Moreover, the technique does not readily inform potential modifications to the data point of interest.

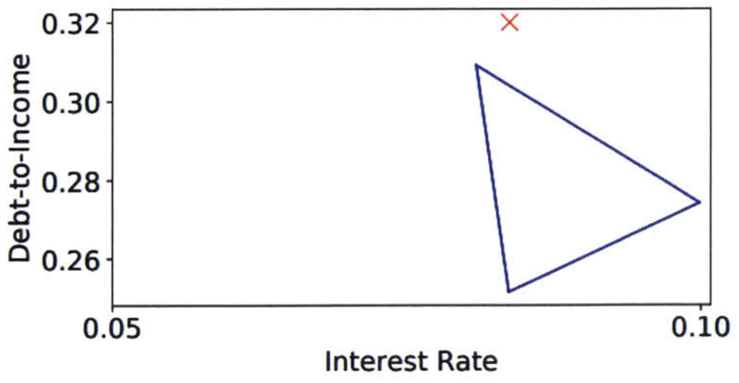

Figure 3-4: Example Correction Generated by Zhang et al. [14]. The red plus denotes an adversely classified point. The blue triangular region represents a minimal, stable, and symbolic correction for that data point. The region inside this triangle is classified favorably.

Suggesting Changes to a Data Point Using Monotonic Guarantees

A function is said to be monotonic with respect to an input variable if it is strictly non-increasing or non-decreasing in changes to that input variable. Although monotonic constraints aid in the global interpretability of a model, their guarantees could be useful for local modifications.

Linear regression models are inherently monotonic because changing the value of a feature will always affect the output prediction in the same direction, depending on the sign of its corresponding coefficient. However, deep learning models do not have this property. Hence, one of the limitations of using linear approximations of deep learning models to inform modifications is that changing feature values may not always change the output variable in the desired direction. One approach to resolve this drawback is to artificially enforce monotonic constraints on neural networks [18].

As an example, Figure 3-6 shows the relationship between the distance of a customer to a cafe and their resulting happiness. Each purple dot represents a customer. The purple line represents a decision boundary learned by a machine learning model trained on this data. Say, we consider it undesirable for our model to predict that beyond the first kilometer, increasing the Distance to Cafe increases User Happiness.

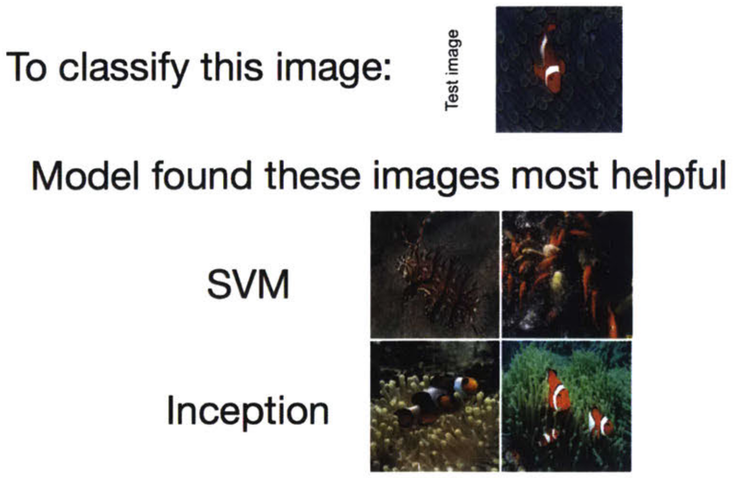

Figure 3-5: Illustration of Results Produced by Influence Functions [17]. The pictures in the 4x4 grid represent training examples that contributed most to the classification of the gold fish in the test example.

For simplicity and ease of understanding, we might want to enforce a monotonic constraint that User Happiness should always be non-increasing with increasing Distance to Cafe. A model that respects this constraint might learn the green decision boundary instead. Hence, if a customer is unhappy, we can advise them with high certainty that moving closer to a cafe will make them feel happier.

Although this approach sounds promising, enforcing such constraints artificially may lead to biased estimators. This, in turn, hurts the performance of our machine learning models.

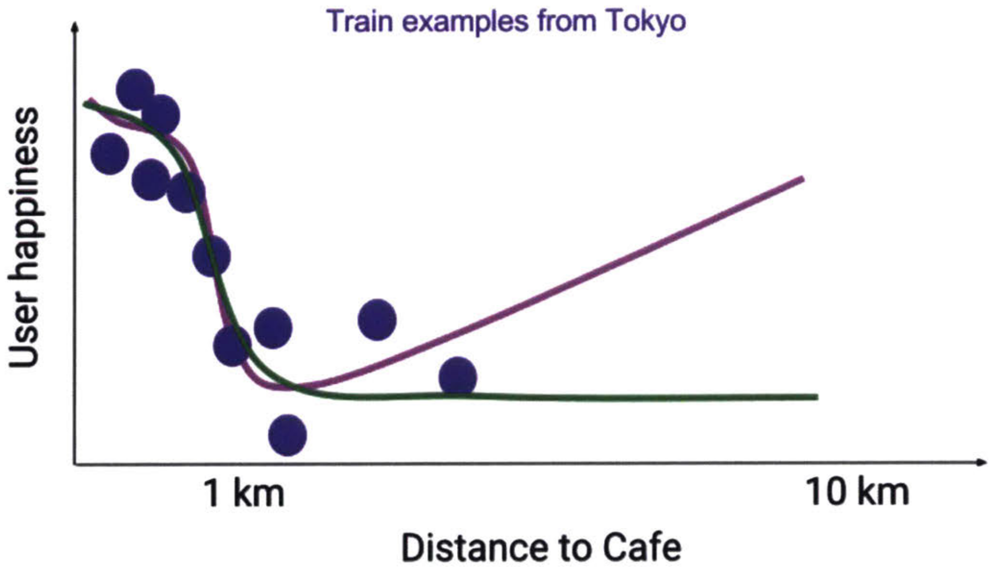

Figure 3-6: Illustration of Enforcing a Monotonic Constraint Between User Happiness and Distance to Cafe [19]. The purple line represents the decision boundary learned from this data. The green line represents the decision boundary learned with the monotonic constraint that User happiness must non-increasing with increasing Distance To Cafe.

Chapter 4

Data

In this chapter, we introduce the credit dataset used in our analyses and describe its 23 predictor features. We also explain our two approaches to dealing with special values in the dataset, namely dropping them or encoding them appropriately. Lastly, we visualize key characteristics of the dataset such as the correlations among features and the results of principal components analysis.

4.1 Description of HELOC Dataset

We used a home equity line of credit (HELOC) dataset provided by FICO. HELOC is a revolving loan where the collateral is the borrower's equity in their house. Similar to a credit card, a HELOC is available for a set time-frame during which a borrower can withdraw money as needed.

4.1.1 Summary and Construction of the Dataset

The dataset contains $10,459$ borrowers who were granted HELOCs during a two-year application window from March 2000 to March 2002. In March 2003, a year after the application window had closed, a performance snapshot was captured and the risks of the borrowers were evaluated. In this dataset, an applicant might have used their HELOC for a duration between one and three years, depending on the timing of their approval. These timelines are illustrated in Figure 4-1.

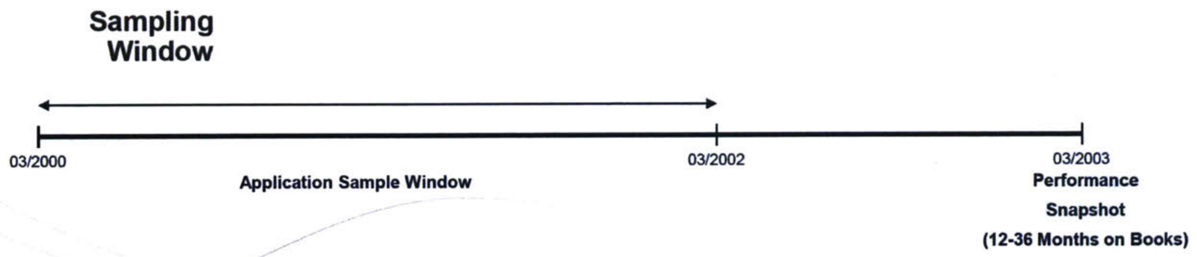

Figure 4-1: Sampling Window of the HELOC Dataset. Applicants were able to use their HELOC for anywhere between one and three years before their performance was evaluated.

Figure 4-2 shows the distribution of Creditworthy and Non-creditworthy classifications. Those with delinquent or charged-off HELOCs were classified as Non-creditworthy (5,459 records) and the rest as Creditworthy (5,000 records). These labels provide the ground truth for the actual behavior of the applicants in the dataset.

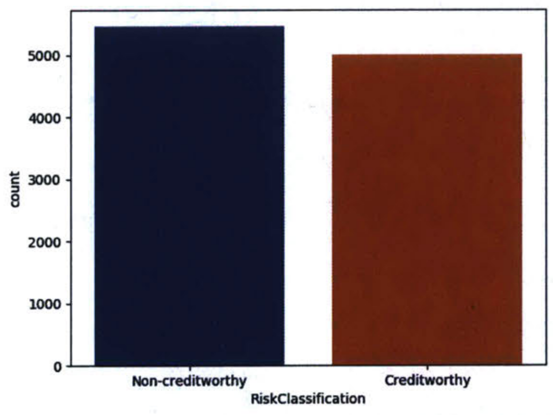

Figure 4-2: Distribution of Risk Classification. The number of Creditworthy and Non-Creditworthy applicants is roughly evenly-split by construction in this dataset.

4.1.2 Glossary of Relevant Credit Modeling Terms

The following definitions provide helpful context for the description of the dataset's features.

1. Line of Credit - An agreement to provide credit.
2. Revolving Line of Credit - A line of credit with a maximum amount that the borrower can choose to use each month. The most common example is a credit card.
3. Installment Line of Credit - A line of credit with a fixed loan amount and a fixed monthly payment. A mortgage is a common example.
4. Delinquent - A line of credit is delinquent if its payments are not made in a timely manner.
5. Utilization - The amount still owed divided by the total amount borrowed; The fraction of available credit currently in use.

4.1.3 Explanation of Predictor Features

In addition the binary target variable (Risk Classification), each credit applicant is characterized by 23 predictor features (21 continuous and 2 categorical). These are:

1. A condensed version of the borrower's credit risk computed by FICO using all credit bureau information (ExternalRiskEstimate)
2. Months since the very first line of credit was established (MSinceFirstLOC)
3. Months since the newest line of credit was established (MSinceNewestLOC)
4. Average age in months of all existing lines of credit (AvgAgeOfLOC)

5. Number of lines of credit not currently delinquent (NumLOCNotDelq)

6. Number of lines of credit ever been 60 or more days delinquent (NumLOC60PlusDaysDelq)

7. Number of lines of credit ever been 90 or more days delinquent (NumLOC90PlusDaysDelq)

8. Percentage of lines of credit never been delinquent (PercentLOCNeverDelq)

9. Number of months since the most recent delinquency (MSinceMRecentDelq)

10. Maximum delinquency in days in the past year (MaxDelqLast12M)

11. Maximum delinquency ever in days (MaxDelqEver)

12. Total number of lines of credit established (NumTotalLOC)

13. Number of lines of credit established in the past year (NumLOCInLast12M)

14. Percentage of lines of credit that are installment lines of credit (PercentInst-LOC)

15. Months since the newest request for a new line of credit excluding those requested in the past week (MSinceNewLOCReqExPastWeek)

16. Number of requests for new lines of credit in the last 6 months (NumLOCReqLast6M)

17. Number of requests for new lines of credit in the last 6 months excluding those requested in the past week (NumLOCReqLast6MExPastWeek)

18. Fraction of all revolving credit limits in use (FracRevLOCLimitUse)

19. Fraction of all installment lines of credit in use (FracInstLOCUse)

20. Number of revolving lines of credit with outstanding balances (NumRevLOCW-Balance)

21. Number of installment lines of credit with outstanding balances (NumInst-LOCWBalance)

22. Number of bank loans and national loans (a subset of all revolving trades) with an outstanding balance of at least 75% of the credit limit (NumBank/NatlLoansWHighUtil)

23. Percentage of lines of credit with outstanding balances (PercentLOCWBalance)

4.1.4 Permissible Values of Categorical Features

The two categorical features in our dataset are the maximum delinquency in days in the past year (MaxDelqLast12M) and the maximum delinquency ever in days (MaxDelqEver). Table 4.1 explains the possible values they can take. Though one might expect these features to be numerical, they are used to indicate the occurrence of certain scenarios in this dataset for which a numerical value would not be meaningful. Notice that subsets of the permissible values have a natural ordering (e.g. C-F) but no ordering extends across the entire set.

Value	Explanation
A	The value is missing
B	There is a derogatory comment in the applicant's public records
C	The applicant has been 120+ days delinquent
D	The applicant has been 90+ days delinquent
E	The applicant has been 60+ days delinquent
F	The applicant has been 30+ days delinquent
G	It is not known if the applicant has been delinquent
H	The applicant has never been delinquent
I	Other

Table 4.1: Explanation of Categorical Features Values for MaxDelqLast12M and MaxDelqEver. Subsets of these values have particular orderings (e.g. C-F) but there is no strict ordering over the entire set.

For instance, if an applicant has filed for bankruptcy in the past year, it will show up as a derogatory comment in their public record and $MaxDelqLast12M = B$. If the applicant has ever had a line of credit that was 120+ days delinquent, then $MaxDelqLast12M = C$.

4.2 Special Values

The dataset contained three types of special values.

-7 : Condition not met. For example, if a credit applicant has never requested for a line of credit or has never been delinquent.

-8 : No usable trades.

-9 : No bureau record or no investigation

Though these values are negative integers, they are interpreted symbolically and do not hold any numeric significance. Therefore, we could not directly feed them into our machine learning models. We either had to drop them or encode them appropriately.

4.2.1 Dropping Special Values

Dropping Data Points That Contained Special Values

A large fraction of the data points in our HELOC dataset contained at least one special value ($7,957$ of the $10,459$ data points). Hence, dropping all such data point was infeasible.

However, the dataset contained 588 records that solely contained the special value -9 as all feature values. Among these, 331 data points were labeled as non-creditworthy and 266 as creditworthy. This was a problem because the same input vector had been given opposite labels. This happens because a borrower receives a -9 special value if they are either a VIP and do not need to be investigated or if they have no bureau record at all i.e. they have no credit history. This confounding of no bureau report

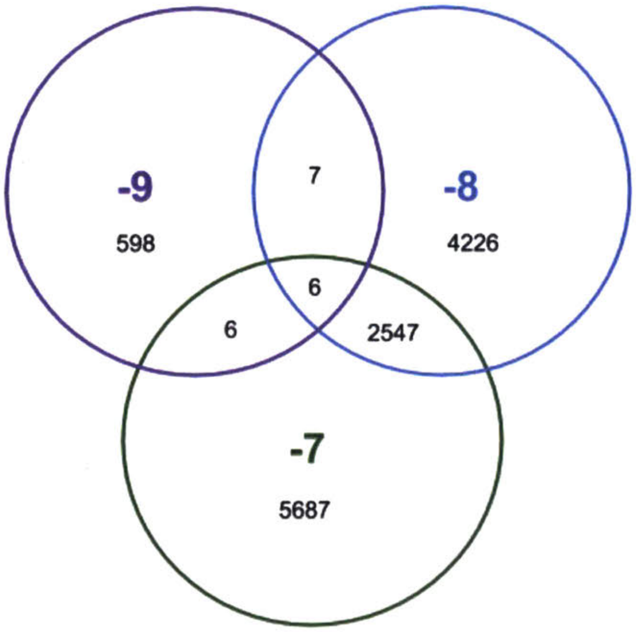

Figure 4-3: Venn Diagram of Records containing at least one special value. 7,957 applicants contain at least one special value. Hence, it is infeasible to drop them all.

investigated (a positive trait for extending credit) and no bureau report found (a negative trait for extending credit) produced the discrepancy in classification. Given that these data points were not meaningful and represented a small fraction of the entire dataset, they were all dropped.

Dropping Features That Contained Special Values

We tabulated the distribution of -7 and -8 special values and found that they are concentrated among 9 of our 23 input features. One way of dealing with these special values is simply dropping the 9 features entirely. Doing so would also not affect the number of points in our dataset. Though this was a valid approach, we found subsequently that dropping the 9 features deteriorated the performance of our machine learning models. The features were likely useful predictors. Therefore, we had to handle the special values in a different way.

Index	Count	-7	-8
1	MSinceFirstLOC	0	239
2	MSinceMRecentDelq	4664	176
3	MSinceNewLOCReqExPastWeek	1855	476
4	FracRevLOCLimitUse	0	186
5	FracInstLOCUse	0	3419
6	NumRevLOCWBalance	0	156
7	NumInstLOCWBalance	0	861
8	NumBank/NatlLoansWHighUtil	0	583
9	PercentLOCWBalance	0	18

Table 4.2: Distribution of Special Values Across Columns. These special values are concentrated among 9 of our 23 input features.

4.2.2 Replacing Special Values with Feature Means

A standard technique for dealing with special values is to replace them with the mean values of the respective features. A simple example illustrates that this would not be a meaningful approach for our dataset.

If a borrower has never had a delinquency, they will have the -7 (Condition not met) special value for the feature MSinceMRecentDelq. Clearly, replacing the feature value with the mean would not be correct as they will be moved from a desirable value of the feature to a less desirable one. An alternative would be to replace the feature value with a very large number, implying it has been a long time since the borrower has had a delinquency. Unfortunately, this comes at the cost of introducing outliers in the dataset.

4.2.3 Binning Features

Another approach to dealing with special values is to discretize continuous features into bins. Consider a sample feature that contains values in the range $[0, \infty)$ and also a special value of -9. As per the binning schema in Table 4.3, every possible value of the feature would map uniquely to a bin. For instance, a value of 3 would map to

Bin 0, a value of 8 would map to Bin 2, and a special value of -9 would map to Bin 3. The advantage of binning is that special values can be treated as a separate bin and any outliers can be consolidated.

Once a binning schema has been decided, a feature can be represented using one-hot encoding and weight-of-evidence encoding (among other options).

Bin Index	Start	End
0	0	4
1	4	8
2	8	∞
3	-9	-9

Table 4.3: Illustration of Binning a Sample Feature. Each feature value maps uniquely to a single bin.

One-hot Encoding

A feature that contains n bins can be represented as an n dimensional vector f. If a feature value belongs to bin_i then the value of its k^{th} dimension $f_k = 1$ if $k == i$ and 0 otherwise, for $k \in [0, n)$. The drawbacks of one-hot encoding are that bins are treated as unordered categories and sparsity is introduced. As per the toy example, the feature value of 8 (mapped to bin 2) would not necessarily be considered greater than the feature value of 3 (mapped to bin 1) because the bins are unordered. Moreover, sparse features could result in overfitting and biased parameters in a model if its training dataset is small. Lastly, for continuous variables, there isn't a clear-cut formula to define the binning schema and choosing it manually could be suboptimal.

Weight of Evidence Encoding

Weight of Evidence (WoE) encoding is a popular statistical technique used in the credit rating industry. It is used to automatically recode the values of continuous and categorical predictor variables into discrete bins and to assign each bin a WoE value.

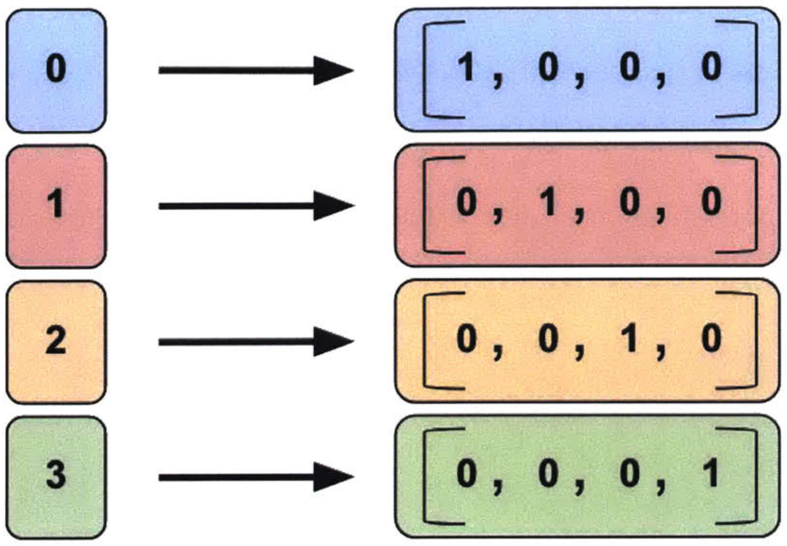

Figure 4-4: Illustration Of One Hot Encoding [20]. A feature with four possible values is mapped into a four dimensional space where exactly one dimension per vector is "hot" i.e., has a unit value.

The bins are determined such that they will produce the largest differences with respect to the WoE values. Additionally, monotonicity constraints can be specified to ensure that WoE values are strictly increasing or decreasing in feature values.

The formula for WoE is derived from entropy theory and information value. For bin_i, the WoE value can be computed as follows:

$$WoE_i = [\ln(\frac{RelativeFrequencyofGoods}{RelativeFrequencyofBads})] * 100 \tag{4.1}$$

Intuitively, the WoE value of a bin provides a measure of its predictive ability to separate creditworthy and non-creditworthy applicants. An important benefit of WoE encoding is that it can be used to treat missing values and outliers without introducing sparsity. As WoE values are on the same scale, we can use them to compare the univariate effects of bins on the target variable within a feature or across all features. Its drawback, like most binning techniques, is that it results in a loss of information.

Table 4.4 shows dummy WoE values for the sample feature. These would be computed from a training dataset using Equation 4.1. As an example, any feature value in bin_0 i.e. in the range $[0, 4)$ is mapped to a WOE value of -0.12.

Bin Index	Start	End	WoE Value
0	0	4	-0.12
1	4	8	0.1
2	8	∞	0.4
3	-9	-9	-0.05

Table 4.4: Weight of Evidence Encoding For A Sample Feature. WoE values are discrete and no two bins share the same WoE value.

Figure 4-5 shows the original distribution of the feature for the number of months since the very first line of credit was established (MSinceFirstLOC) in our dataset. Once the feature was re-coded using WoE, its values were discretized as per Figure 4-6.

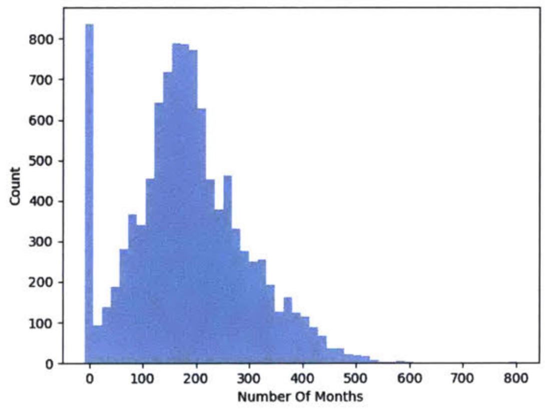

Figure 4-5: Histogram of Feature - Months since the very first line of credit was established (MSinceFirstLOC). The feature is approximately normally distributed and we see a large concentration of values at 0.

As described in Chapter 5, we found that models trained on WoE encoded data had the best performance. Therefore, this technique was chosen to deal with the special values.

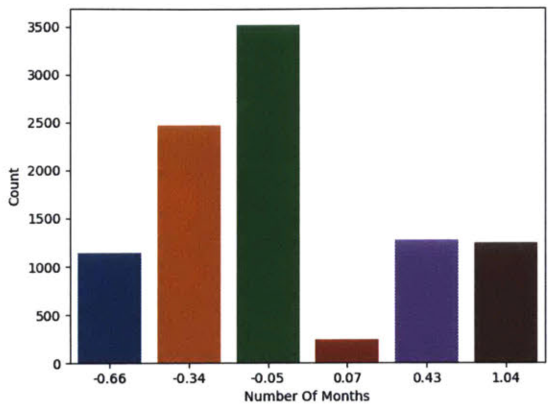

Figure 4-6: Countplot of WoE Encoded Feature - Months since the very first line of credit was established (MSinceFirstLOC). The continuous values of the feature in Figure 4-5 are discretized into WoE values.

4.3 Exploratory Analysis

Before we used the dataset to train machine learning models, we analyzed its properties using a few exploratory visualization techniques.

4.3.1 Correlation of Features

Figure 4-7 visualizes the 23x23 correlation matrix of our dataset and identifies higher correlations with lighter shades. We found three pairs of features with correlations greater than 0.8.

1. The total number of lines of credit (NumTotalLOC) and the number of lines of credit that are not currently delinquent (NumLOCNotDelq)
2. The number of lines of credit that have been 60+ days delinquent (NumLOC60PlusDaysDelq) and the number of lines of credit that have been 90+ days delinquent (NumLOC90PlusDaysDelq)
3. The number of request for new lines of credit in the past 6 months (Num-

LOCReqLast6M) and the number of request for new lines of credit in the past 6 months excluding the past week (NumLOCReqLast6MExPastWeek)

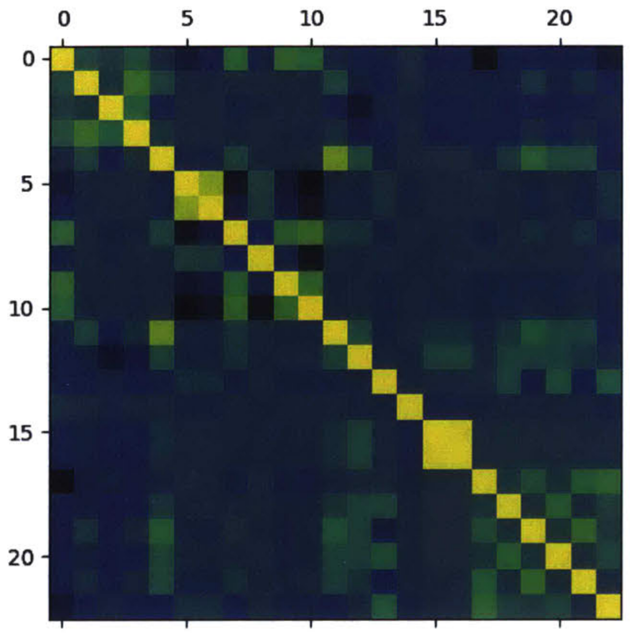

Figure 4-7: 23 x 23 Correlation Matrix. The lighter shades represents highly correlated feature pairs. As expected, the diagonal has the lightest shade because it represents the correlation of a feature with itself.

4.3.2 Principal Component Analysis

Principal Component Analysis (PCA) is a commonly-used statistical procedure that converts a set of possibly correlated variables into a new set of linearly uncorrelated variables called principal components.

The top three principal components were visualized in Figures 4-8, 4-9, 4-10. It can be seen that the Creditworthy (green dots) and the Non-creditworthy (red dots) samples are not readily separable. The three dimensions represent the top three directions of the maximal variance of the dataset but are a result of linear combinations of the original features and thus do not have a human-interpretable meaning.

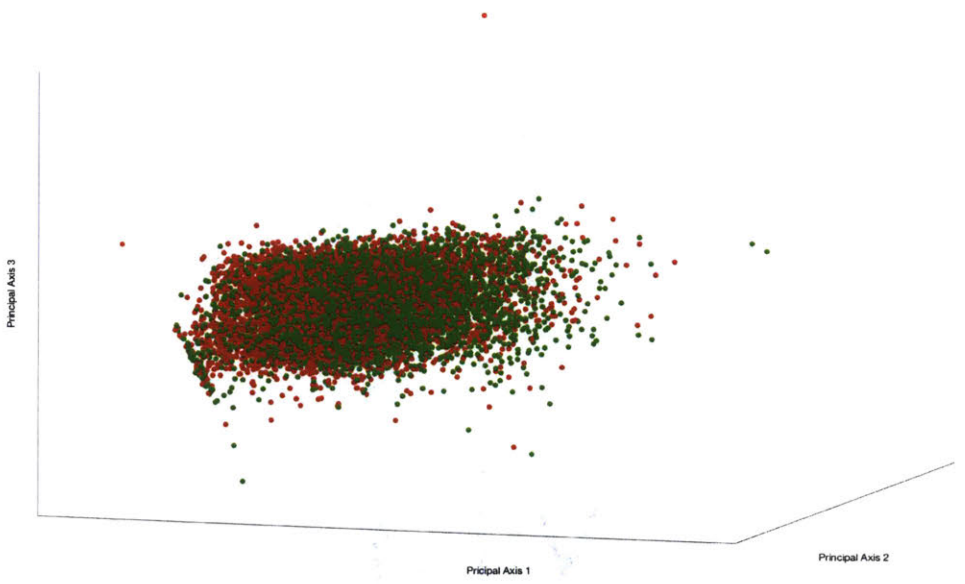

Figure 4-8: 3D Principal Components Analysis (1 of 3). The green and red dots are not readily separable.

In Figure 4-11, we vary the number of principal components and plot the cumulative explained variance. We see that 15 principal components are sufficient to represent 95% of the variance of the 23-dimensional dataset.

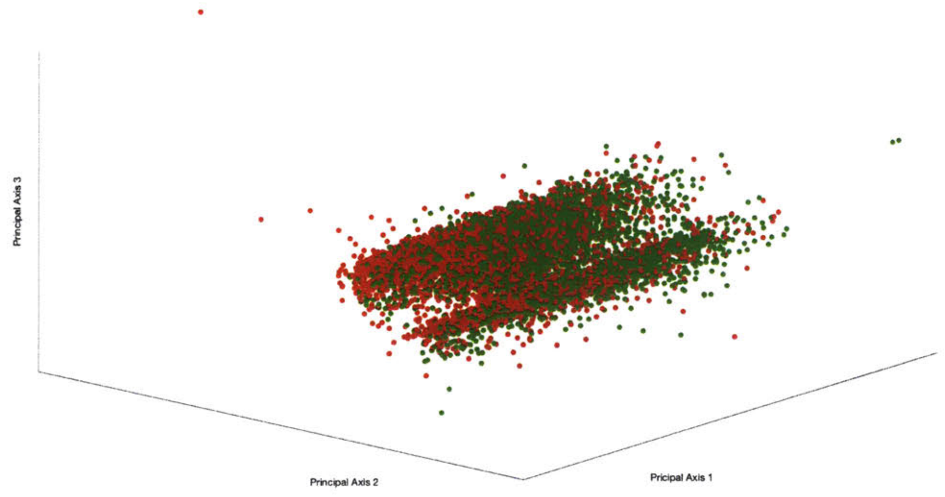

Figure 4-9: 3D Principal Components Analysis (2 of 3). The green and red dots are not easily separable.

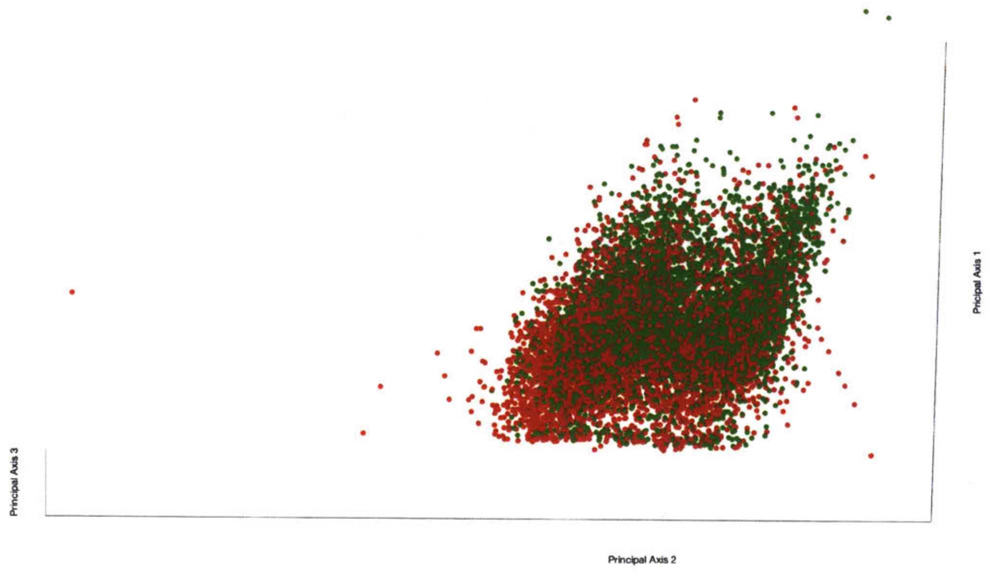

Figure 4-10: 3D Principal Components Analysis (3 of 3). The green and red dots are not easily separable.

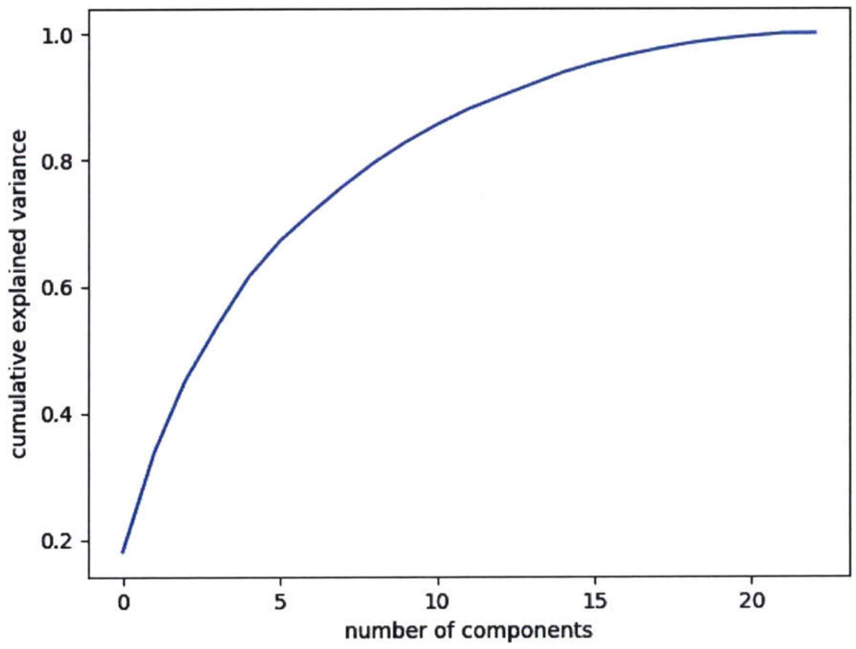

Figure 4-11: Explained Variance vs Number of Principal Components. We see that 15 principal components are sufficient to represent 95% of the variance of the 23-dimensional dataset.

Chapter 5

Machine Learning Algorithms

In this chapter, we explain the theory behind three supervised machine learning models and benchmark their performance on the HELOC dataset. The models, Logistic Regression (LR), Random Forest (RF), and Multi-Layer Perceptron (MLP), will serve as the foundational test-bed on which the interpretability work will proceed.

LR is the most commonly used model in the credit rating industry and thus serves as a useful benchmark. RF is a popular ensemble learning approach that has demonstrated promising performance in credit scoring with highly correlated data [21]. Unlike LR and MLP, RF is a non-differentiable model and thus cannot be approached with gradient-based interpretability techniques. MLP is a class of feed-forward deep neural networks that can learn non-linear relationships without manual feature engineering.

It is possible to compute a global ordering of features for LR and RF with the caveat that it is not always reliable. In contrast, it is not possible to directly compute such an ordering for MLP. Thus, we instead turn to generate a local feature ordering in the next chapter.

5.1 Performance Summary

We are interested in predicting the likely outcome for a HELOC applicant, namely a timely pay-off or a delinquency. Formally, the model must predict $\Pr(Y = 1|X)$ where Y is the binary output variable and X is the input vector of features.

We train the three models on two versions of the HELOC dataset, termed Truncated data and WoE data. The Truncated data contains 14 features and is constructed by dropping all features that contain one or more special values for any of the points in the dataset. WoE data contains 23 features and is constructed by re-coding all features using Weight of Evidence (WoE) encoding.

We evaluate the models using K-fold cross-validation. We partition the input dataset into a training set, which the model trains on, and a test set, which the model is evaluated on. The performance of a model on a test set i.e., samples it has never encountered before, provides a measure of its generalizability. In K-fold cross-validation, the input dataset is randomly partitioned into k equal subsets and in each run, one of the subsets is chosen as the test set and the remaining as the training set. For this analysis, we choose $k = 5$ i.e., in each run of cross-validation, the training set contains 80% of the dataset ($7,898$ points) and the test set contains the remaining 20% ($1,973$ points).

As Table 5.1 illustrates, we find that all three models perform better on the WoE data as compared to the Truncated data. This implies that the loss in information resulting from re-coding the features using WoE encoding is smaller than that resulting from dropping all features with special values. Among the models trained on the WoE data, MLP has the highest mean test accuracy of 74.7%. Interestingly, the same MLP architecture has the worst test accuracy on the Truncated data, possibly because the Truncated data has too few features, given the depth of the MLP architecture. Although we benchmark the performance of RF, we focus our subsequent interpretability work on the best-performing MLP due to its superior accuracy and our interest in interpreting deep learning models.

Dataset	Fold	LR		RF		MLP	
		Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
Truncated	Train	0.724	0.002	0.99	0	0.666	0.004
Truncated	Test	0.723	0.008	0.706	0.007	0.664	0.005
WoE	Train	0.746	0.002	0.99	0	0.749	0.001
WoE	Test	0.744	0.010	0.723	0.007	0.747	0.008

Table 5.1: 5-Fold Cross-Validation Performance of Machine Learning Models Trained On Truncated and WoE Data. The MLP trained on WoE data had the best mean test accuracy of 74.7%.

5.2 Model

5.2.1 Logistic Regression

Logistic Regression has historically been the industry standard for credit scoring. It is commonly used for binary classification tasks and is therefore well-suited to distinguishing between Creditworthy and Non-creditworthy applicants.

Theory

As seen in Figure 5-1, logistic regression differs from traditional linear regression in that its output is constrained to be between 0 and 1. This property gives its output a useful interpretation - the probability of a binary outcome event. A logistic regression model is constructed by passing in a linear combination of input features through a standard logistic function (sigmoid function), which takes any real input, and outputs a value between zero and one. This can be represented mathematically as:

$$\Pr(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 \cdot X_1 + + \beta_2 \cdot X_2 + \ldots + \beta_k \cdot X_k)}}$$

where $X_1 \ldots X_k$ are the input features and $\beta_0 \ldots \beta_k$ are the regression coefficients.

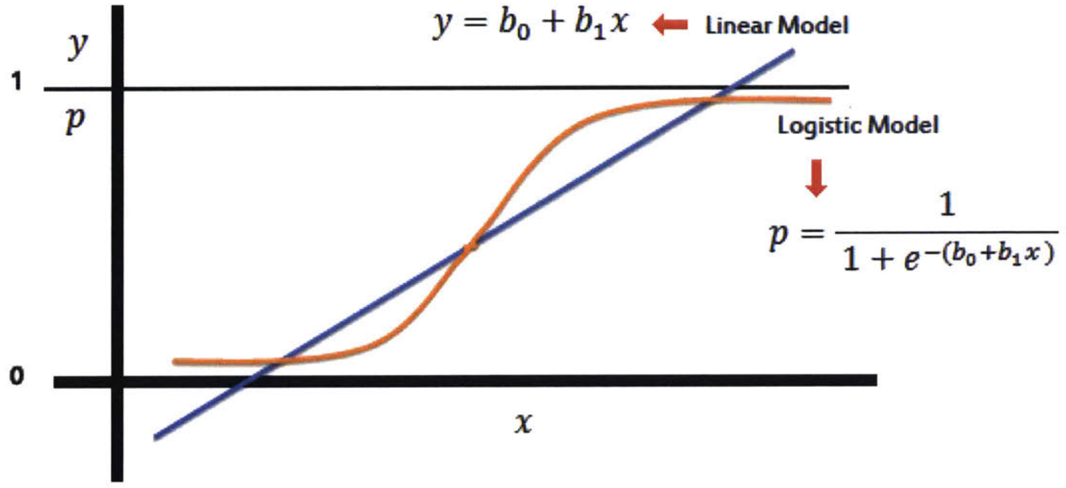

Figure 5-1: Comparison of Decision Boundary Between Logistic Regression and Linear Regression. Unlike linear regression, the output of logistic regression is constrained to be between 0 and 1.

The β's i.e., the coefficients of logistic regression are computed using the Maximum Likelihood Estimator, which is a method of estimating the parameters of a statistical model, given observations.

Global Interpretation

The interpretation of the regression coefficients of logistic regression is not as straightforward as that of the regression coefficients of linear regression. By re-arranging the equation of logistic regression, it can be seen that the log-odds of $Pr(Y = 1|X)$ is a linear combination of the input features.

$$\log[\frac{Pr(Y=1|X)}{1-Pr(Y=1|X)}] = \beta_0 + \beta_1 \cdot X_1 + \beta_2 \cdot X_2 + ... + +\beta_k \cdot X_k$$

β_i can be interpreted as the change in log-odds resulting from a unit change in X_i, given that the other features are held constant.

If we would like to determine the relative importance of features in a logistic regression, we cannot directly rely on the coefficients of a model trained on raw data. This is because the features may have different scales. To correct for this, we could normalize the values of each feature across the dataset by giving it a zero mean and a unit standard deviation. Once a logistic regression is trained on this standardized data, the absolute values of its coefficients can be used as a measure of relative feature importance. Figure 5-2 shows the relative feature importance computed from a logistic regression trained on standardized Truncated data.

One of the shortcomings of this approach is that it is constrained by the assumption in logistic regression that features are uncorrelated. If this assumption does not hold, the approach may provide an imperfect measure of the relative feature importance.

5.2.2 Random Forest

Random Forest is an ensemble supervised learning technique i.e., it consists of an aggregation of multiple outputs made by a diverse set of predictors, in this case, multiple decision trees, in the belief that this will produce a more accurate result.

Theory

The key idea behind Random Forest is that a high-performance classifier can be constructed from a set of poorly-performing classifiers. This is similar to the idea

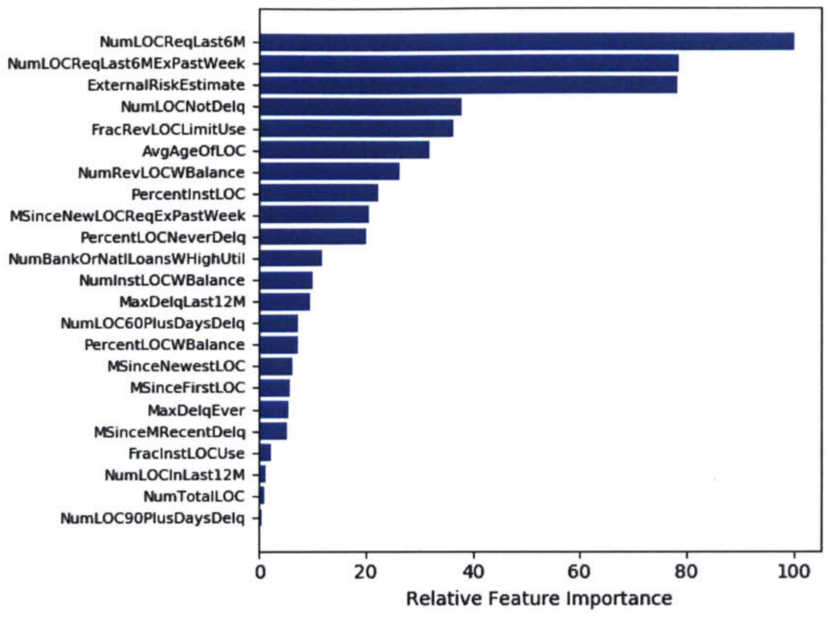

Figure 5-2: Relative Feature Importance Computed from Logistic Regression. The regression is trained on scaled Truncated Data so that coefficient values can be directly compared. We see that the number of requests for new lines of credit in the past 6 months is identified as the most significant feature.

that a team can reach a better decision than any member can individually. Random Forest computes a prediction for the output variable by averaging the predictions of numerous decision trees (weak classifiers).

A (single) decision tree is a supervised learning method that predicts the value of a target variable by learning simple If-Then decision rules. It is constructed using the Classification And Regression Tree (CART) algorithm. Each node in the decision tree is a condition on the value of a single feature that splits the data into two subsequent branches. CART recursively identifies the feature-value pair that best minimizes the tree's Gini impurity.

Gini impurity is a metric of the disorderliness of the labels of a set of data points. Given, a dataset X with binary labels, its Gini impurity is given by the formula:

$$G(X) = 1 - \Pr(Y = 0)^2 - \Pr(Y = 1)^2$$

where $\Pr(Y = 0)$ and $\Pr(Y = 1)$ are computed using the relative frequencies of the two labels.

In the best case, if the points contain only one label, then either $\Pr(Y = 0) = 1$ or $\Pr(Y = 1) = 1$ and $G(X) = 0$. In the worst case, there is an even split between the two labels and $G(X) = 0.5$. When the CART algorithm considers a new node in a decision tree, it computes the Gini impurity for each of the two resulting groups of the dataset and averages them, weighted by the number of points in each group. The feature-value pair that best minimizes this weighted value is chosen.

The drawback of individual decision trees is that they are susceptible to overfitting and do not generalize well i.e., they have high variance. In fact, drawing different samples from the same population could produce decision trees with vastly different nodes and performances. Random Forest overcomes this shortcoming by averaging the outputs of a collection of trees.

As illustrated in Figure 5-3, Random Forests are trained via a method called bootstrap aggregation or bagging. The training data points are first randomly assignment into n groups with replacement, where n corresponds to the number of decision trees. In this example, $n = 4$. Individual decision trees are fit on a randomly chosen set of features in each group. To classify a new data point, the Random Forest aggregates the predictions from each of its constituent decision trees and outputs the majority vote as its classification. In this example, the point receives a majority classification of Class C. The random sampling of data points and features ensures that the resulting decision trees are uncorrelated. Thus, by aggregating their independent predictions, Random Forest is able to reduce variance and improve generalizability.

Figure 5-4 shows that Random Forest models trained on WoE data achieve a maximum test accuracy of 73.2% at $n \geq 30$ decision trees.

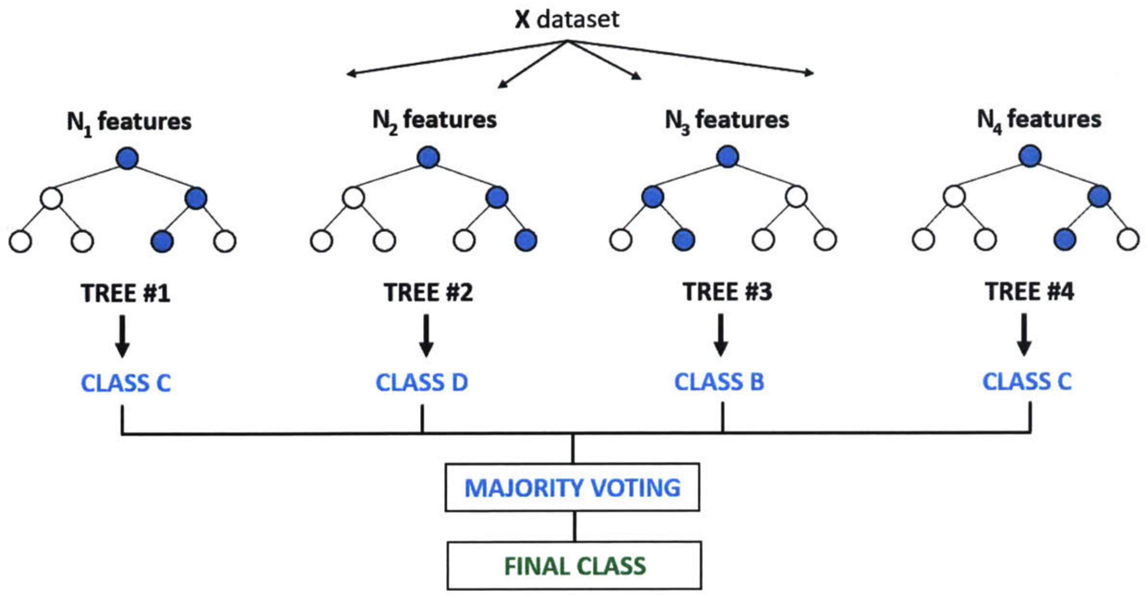

Figure 5-3: Illustration of Random Forest. To classify a new data point, the Random Forest aggregates the predictions from each of its constituent decision trees and outputs the majority vote as its classification. In this example, the point receives a majority classification of Class C.

Global Interpretation

The CART algorithm computes how much each feature decreases the Gini impurity in a tree. This number can be averaged for each feature across all trees of a Random Forest to compute a global feature importance ranking. The more a feature decreases the Gini impurity, the more important it is. Figure 5-5 illustrates the ranking generated by Random Forest on the WoE encoded HELOC data.

As was the case for Logistic Regression, the feature importance ranking computed by Random Forest is affected by the presence of correlated features. For instance, let there be two heavily correlated features of similar predictive power. Using one feature in a decision tree would dampen the ability of the other feature to decrease the Gini impurity when it is considered as a potential node. This would lead to a lower relative importance for the second feature and lead to incorrect conclusions

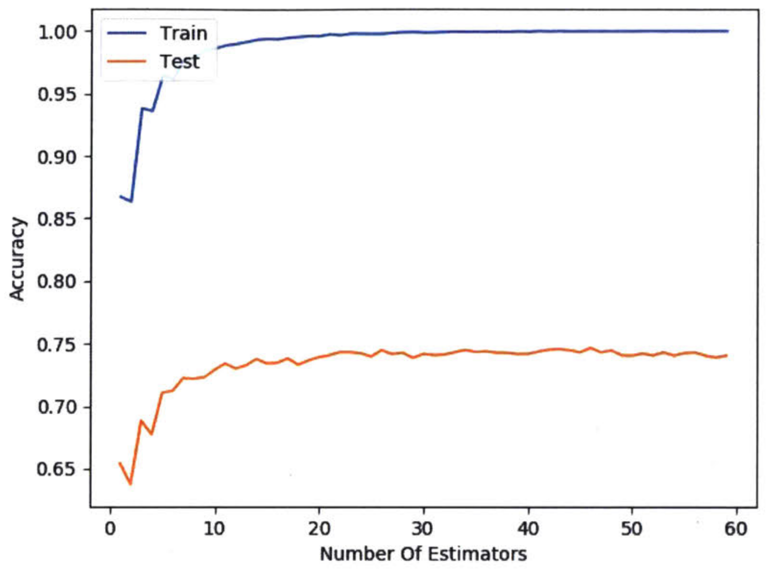

Figure 5-4: Accuracy of Random Forest vs Number of Estimators on WoE Data. The maximum accuracy is achieved once there are at least 30 decision trees.

about the predictive power of features.

5.2.3 Multi-Layer Perceptron

A multi-layer perceptron (MLP) is an artificial neural network with one or more hidden layers. Neural networks are supervised learning models loosely inspired by the biological networks of the human brain.

Theory

An MLP is composed of three types of layers - an input layer, hidden layers, and an output layer. An input layer relays the input features into the model, the hidden layers act as the computational engine, and the output layer generates the final model prediction.

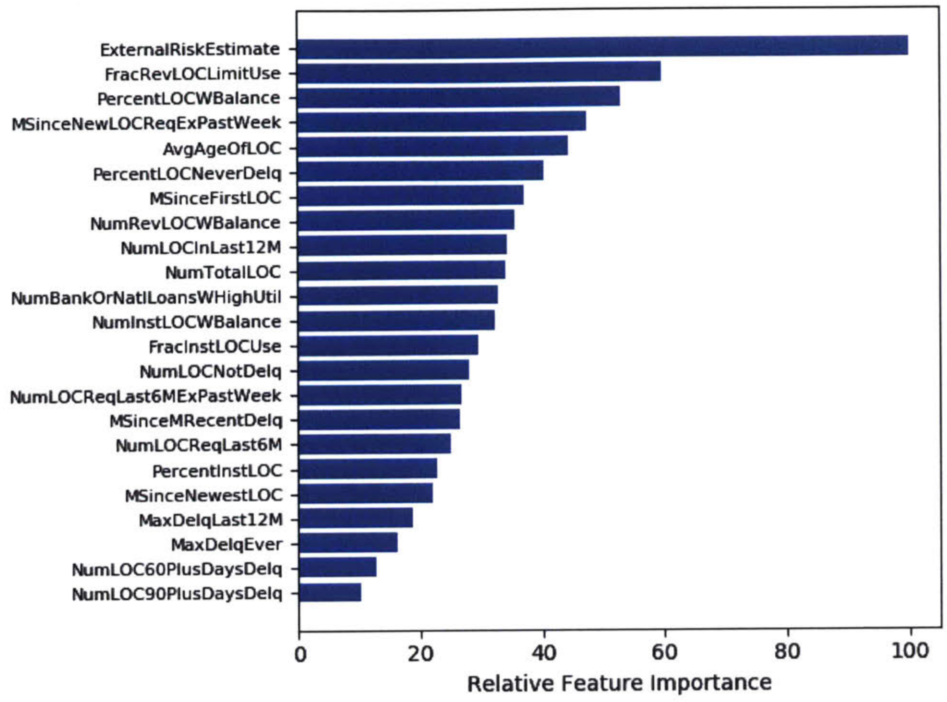

Figure 5-5: Relative Global Feature Importance for Random Forest. This ordering is determined by measuring the average decrease in Gini impurity produced by each feature across all decision trees.

Each layer of a neural network is made up of units called neurons. Figure 5-6 shows an example MLP architecture. It consists of an input layer with three neurons, a hidden layer with three neurons, and an output layer with one neuron.

The input to each hidden unit is a linear combination of the units of the preceding layer. The hidden unit then computes an output by mapping its input through an activation function. A non-linear activation function such as a Rectified Linear Unit (ReLU) is commonly used to create non-linear interactions between the units of the neural network. It is worth noting that logistic regression is a special case of an MLP with one hidden layer containing one hidden unit with a sigmoid activation function.

As illustrated in Figure 5-7, for an input value u,

$$ReLU(z) = \max(0, z)$$

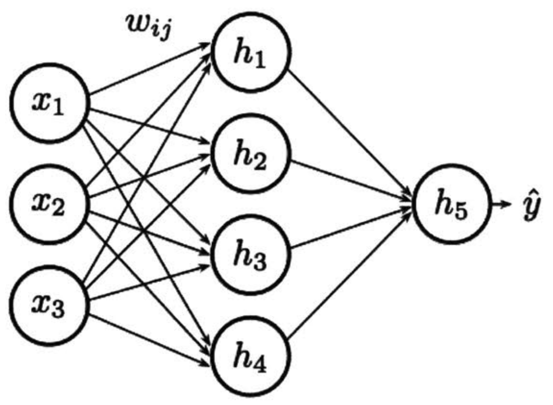

Figure 5-6: A Multi-Layer Perceptron [22]. It has an input layer with three neurons, one hidden layer with four neurons, and an output layer with one neuron.

In Figure 5-6, the value of each hidden unit can be computed as:

$$h_j(x) = ReLU(w_j + \sum_{i=0}^{n} w_{ij} \cdot x_i)$$

Here, w_{ij} is the weight from input x_i to hidden unit h_j. These weights are the parameters of the neural network and are trained by an optimization technique called back-propagation. As with any machine learning model, training neural networks entails minimizing misclassifications of training data with the goal of strong performance in the test data. These errors are represented as a differentiable loss function that takes in weights of the neural network as input. Optimizers like Stochastic Gradient Descent (SGD) are used to climb down the loss function and update the weights of the neural network to reduce their contribution to the loss. In theory, SGD only guarantees to reach a local minima of loss, not a global minimum. Despite this, neural networks have reached impressive accuracies with very low losses.

From a theoretical standpoint, there is still a lot to learn about what truly makes neural networks work so well. In particular, choosing the number of layers, the number of units, and the types of activation functions remains a delicate manual exercise.

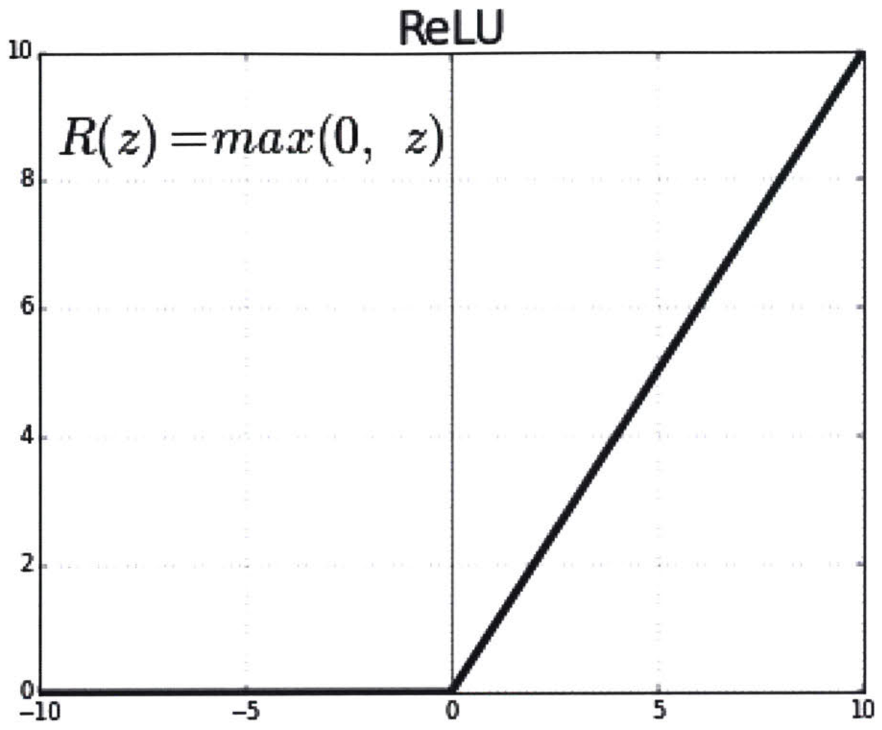

Figure 5-7: ReLU(z) = max(0,z) [23]. The function maps any positive integer to itself and any negative integer to zero.

In Table 5.2, we describe the mean train and test accuracies achieved by several MLP architectures trained on WoE data and evaluated using 5-fold cross-validation. An architecture (a,b,c) represents three hidden layers with a, b, and c hidden neurons respectively. We see a general increase in the train accuracy as we increase the depth of the architectures. On the other hand, test accuracy improves up till (3,2,2) but then decreases. This suggests that the training dataset from FICO may not be large enough to train larger architectures. We chose the (3,2,2) architecture as the best-performing architecture based on mean test accuracy.

Global Interpretation

The MLP that had the best mean test accuracy on the WoE dataset had a (3,2,2) architecture i.e., an input layer with 23 units (for each of the features), one hidden layer with 3 units, two hidden layers with 2 units each, and an output layer with

Hidden Layers Architecture	Mean Train Accuracy	Mean Train Std. Dev.	Mean Test Accuracy	Mean Test Std. Dev.
(3,1)	0.748	0.002	0.742	0.008
(2,3)	0.748	0.002	0.744	0.009
(3,3)	0.748	0.002	0.744	0.012
(5,3)	0.750	0.002	0.743	0.006
(3,2,2)	0.748	0.002	0.747	0.009
(5,5,5)	0.750	0.003	0.743	0.009
(10,10,10,10)	0.757	0.003	0.744	0.008

Table 5.2: 5-Fold Cross-Validation Performance of MLP Architectures on WoE Data. The architecture with the best mean test performance was (3,2,2). We see that the performance of MLP does not improve with deeper architectures beyond a threshold, suggesting that the HELOC dataset may not be suitable for training a large number of parameters.

2 units (one for each of the binary outcomes). Therefore, in total, this model has $23 * 3 * 2 * 2 = 552$ parameters. These are the weights of the network. In contrast, a logistic regression model trained on WoE data only has 22 parameters.

The weights of the neural network create complex, non-linear interactions between input features and do not have human-interpretable meanings. Moreover, the decision-boundaries learned by neural networks can be both high-dimensional and extremely non-linear. Thus, features may have varying significance in different points of the feature space. To our best knowledge, there are no established techniques to generate global feature importance rankings for MLPs.

Chapter 6

Evaluating LIME, a Local Model-Approximation Technique

We begin by motivating our choice of approximating a machine learning model using piece-wise linear local approximations. We explain how LIME, a local model-approximation technique, can be used to generate such approximations. We introduce the original algorithm of LIME and detail its shortcomings. Then, we describe how we intend to address some of these using a modified version of the algorithm. We evaluate both the original and modified implementations on two criteria - Fidelity and Modification Power. We show that the modified implementation outperforms the original one in certain scenarios.

6.1 Motivation for Approximating a Model Using Piece-Wise Linear Local Approximations

Recall that we are interested in answering two questions about the prediction of a machine learning model for a sample data point. Why did the data point receive the classification that it did? What small changes can be made to the feature values of the data point such that its classification reverses?

These tasks can be written more precisely as:

1. **Rationales** - We want to identify the top k features that contributed most to a classification, where k is a parameter selectable by a user.
2. **Effective Modifications** - We want to specify a set of minimal changes to the feature values of a sample point such that its classification reverses.

We view simplicity as a key element of interpretability, motivating one approach to interpreting a black-box machine learning model: approximate it with a simpler model. Though simplicity is, in turn, broad and subjective, we focus on one of its aspects, namely decomposability. Each parameter of a decomposable model admits a human-understandable meaning [24]. For instance, a linear regression is decomposable. Each of its coefficients can be understood as the change in the output prediction caused by a unit change in the corresponding feature value, given that other feature values are held constant. The sum of these coefficients weighted by the feature values yields the linear regression's prediction. In contrast, if a neural network has a complex and non-linear decision boundary, the contribution of its input features towards a prediction will depend on their interactions with other feature values.

It may not be particularly useful to approximate a complex non-linear decision boundary by a single simpler linear model as performance is likely to deteriorate. Instead, we approximate a model piece-wise using several linear models: The key idea is that every segment of the decision boundary starts to look linear the more one zooms in.

LIME can be used to compute such piece-wise linear approximations of an underlying MLP. We evaluate the approximations on two criteria:

1. **Fidelity** - The ability to match the predictions of the model being approximated.

2. **Modification Power** - The ability to inform changes to a data point so that it may cross over from one side of that model's decision boundary to the other.

6.2 Local Interpretable Model-Agnostic Explanations (LIME)

LIME is a model-agnostic technique that can approximate the decision boundary of a model at a particular data point. The approximation is constructed by training a locally-weighted linear regression model in the neighborhood of the data point of interest. The coefficients of the regression can be used to justify the data point's classification and advise how that data point can be modified to reverse its classification.

Figure 6-1 visualizes the coefficients of a sample linear regression generated by LIME. This regression approximates an MLP in the neighborhood of a particular credit applicant and can be used to predict the probability of default of that applicant. In a linear regression, each coefficient can be interpreted as the change in the output produced by a unit change in the corresponding feature value, given that other feature values are held constant. In this figure, a unit change in the values of features with coefficients in red and slanted lines will increase the probability of default. A similar change for features with coefficients in dotted green will decrease the probability of default.

It is worth clarifying three key points:

1. In our work, the LIME regressions are trained on WoE data because our best-performing MLP is trained on WoE data.
2. In a linear regression, a feature value may contribute to the output prediction in a different direction than the sign of its regression coefficient. For instance, if the feature value for a sample point is negative and the regression coefficient

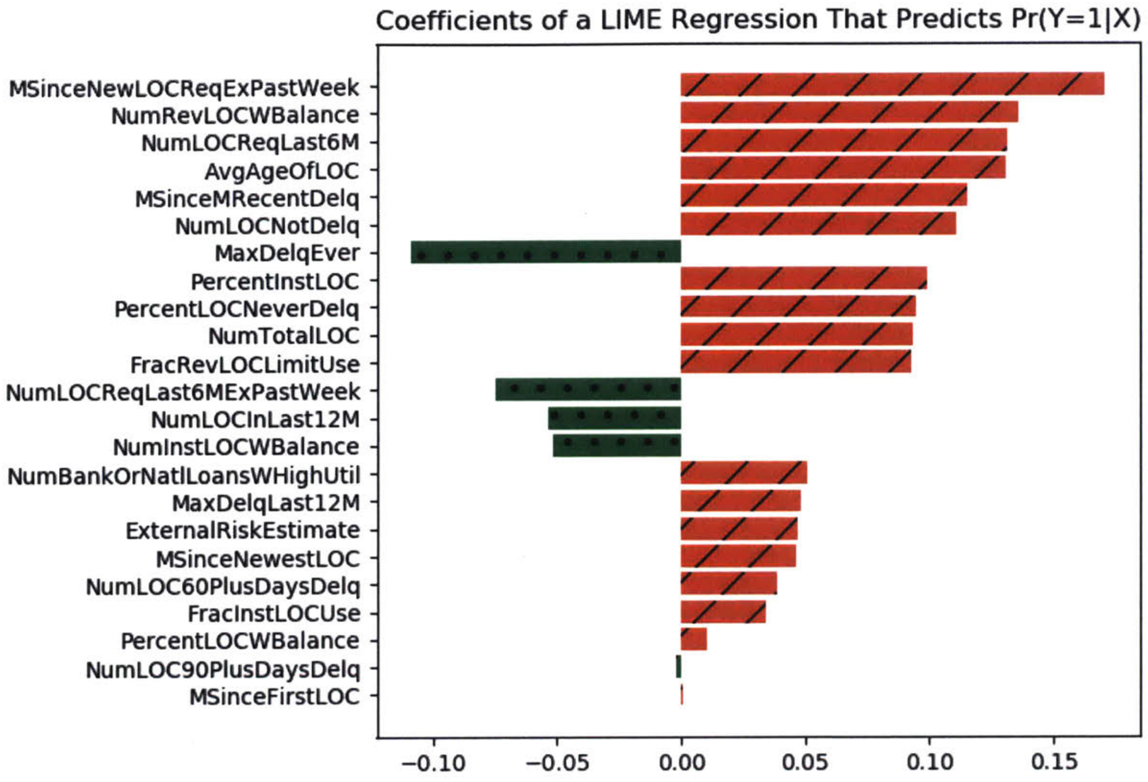

Figure 6-1: A LIME Linear Approximation of an MLP at a WoE-encoded Sample Credit Applicant. Each bar is a coefficient in the linear regression. A unit change in the value of a feature with a bar in red and slanted lines will increase the probability of default. A similar change in the value of a feature with a bar in dotted green will decrease that probability.

for that feature is positive, then that feature will contribute negatively towards the output prediction of that sample point. Similarly, this is true for the linear regressions computed by LIME that are trained on WoE data.

3. WoE encoding discretizes the value of a continuous feature into distinct bins that are each assigned unique WoE values. In this thesis, these WoE values are chosen to be monotonic in the original feature values, either strictly increasing or strictly decreasing. Thus, a unit increase in the WoE value might correspond to either an increase or a decrease in the original feature value.

Through these observations, we can see that the coefficients of a LIME approximation alone are not sufficient to determine the contribution of features towards the prediction of a sample data point. We also require the feature values of that data point.

6.2.1 Original Implementation of LIME

Given a trained model and a data point, the original implementation of LIME generates a local approximation by perturbing the data point and finding the best-fit locally-weighted linear regression. The algorithm is detailed below.

1. The mean and covariance of each feature in the training data are computed.
2. An input parameter specifies whether the normal distribution for each feature is centered at that feature's mean or at the data point of interest. Centering at the data point of interest may not be desirable if the data point is an outlier.
3. A new perturbed data point is sampled by sampling the values of each of its feature independently from univariate normal distributions selected in Step 2.
4. Step 3 is repeated n times to generate n perturbed points. The original implementation chooses $n = 5,000$ (an empirical choice) by default.
5. The trained model predicts the $\Pr(Y = 1|X)$ for each of the perturbed points.
6. The Euclidean distance between each perturbed point and the input data point is computed. These distances are passed into a kernel function that normalizes them into the range $[0, 1]$. The normalized distances represent how to weight each of the perturbed points. The ones closer to the input data point will have higher weights than the ones further away.
7. A linear regression is trained on the weighted perturbed points to predict $\Pr(Y = 1|X)$. The local weighting ensures that the regression is more accurate for perturbed points closer to the input data point.
8. This regression is the linear approximation generated by LIME.

We adapt the original implementation of LIME for our use case. In particular, once the perturbed points are generated in Step 4, we re-code them using Weight of Evidence (WoE) encoding. This is necessary because the MLP chosen as the underlying model for the approximations is trained on WoE data. This model is used to classify the perturbed points (as described in Step 5) and hence, in our particular case, the perturbed points need to be in the WoE space. We choose not to sample perturbed points directly in the WoE space because WoE encoded features are discrete and thus unsuitable for being sampled from normal distributions as per the LIME algorithm. In contrast, we can sample original feature values from normal distributions because they are continuous and approximately normally-distributed.

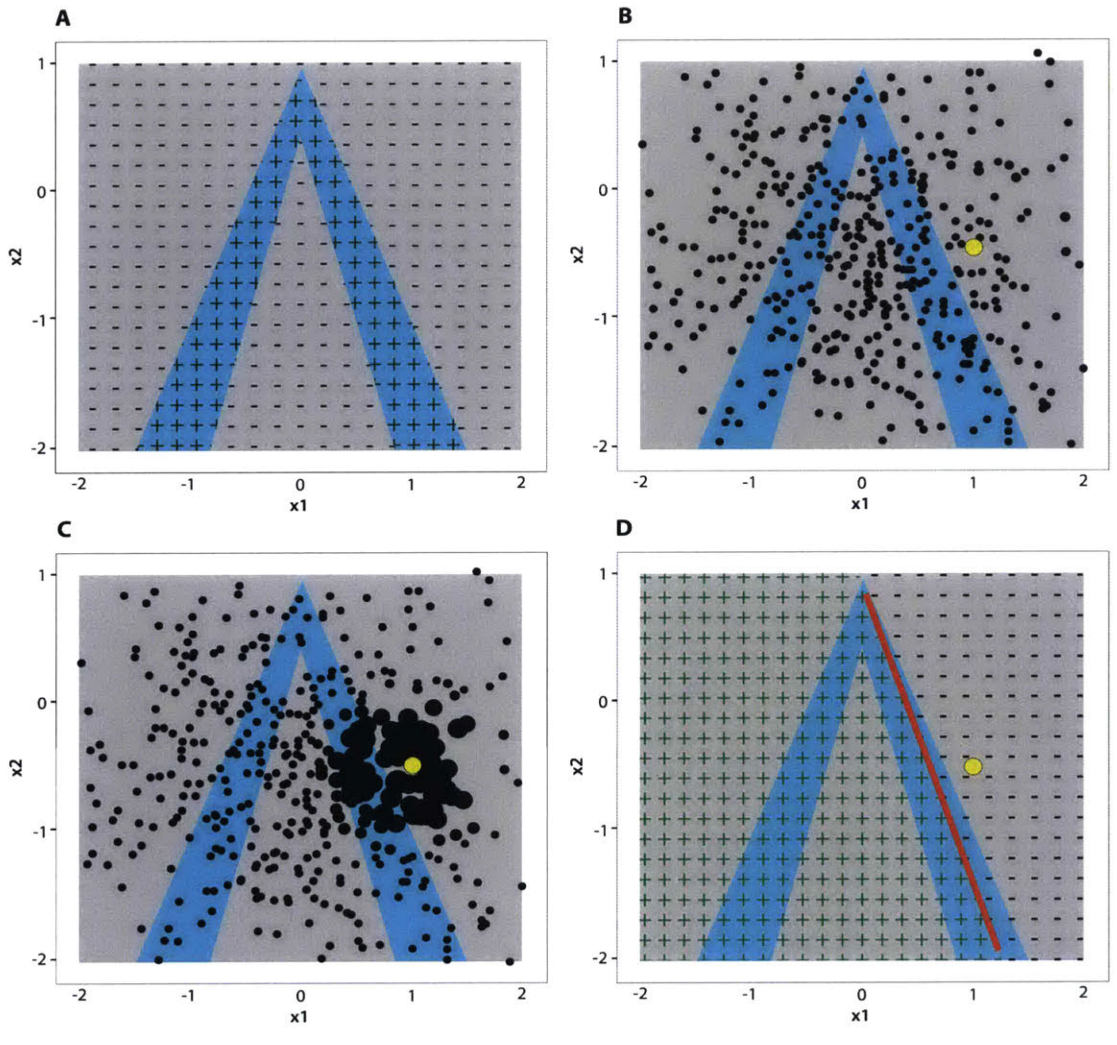

Figure 6-2: Illustration of LIME Algorithm In 2 Dimensions For Simplicity.

Figure 6-2 illustrates how LIME works, using a 2-dimensional space for simplicity. 6-2 (A) shows the decision boundary of a black-box model. The points inside the blue inverted V space have a positive label and the ones outside have a negative label. In 6-2 (B), the yellow point is the data point at which the model must be approximated. The black dots represent perturbed points sampled from normal distributions of the two features. In 6-2 (C), the black dots near the yellow point are given higher weights. The red line in 6-2 (D) describes the linear approximation computed by LIME. The points to its left are classified as positive and those to its right as negative.

6.2.2 Shortcomings of LIME

Recall that LIME assumes features are independent. Violations of this assumption can affect the approximations produced by LIME in two ways:

1. **The values of the regression coefficients cannot be interpreted independently.**
   Figure 6-1 illustrates this point. Notice that the feature for number of requests for lines of credit in the last six months (NumLOCReqLast6M) has a coefficient with a different direction (red and slanted lines vs dotted green) than that of its closely related feature for number of requests for lines of credit in the last six months *excluding the past week* (NumLOCReqLast6MExPastWeek). This is a non-intuitive result because the features have very similar meanings and hence should affect the probability of default in the same direction.

   Such discrepancies are common in any linear regression trained on data with multicollinearity. Although the predictive accuracy of the regression is not affected by this condition, it may not be reliable to interpret the regression coefficients independently. In this thesis, we do not address this general shortcoming of linear regression and its impact on LIME. We suggest alternate regression techniques that are robust to highly correlated features in Section 8.2: Further

Work.

2. **The perturbed data points sampled by LIME may be invalid.**

   This can be illustrated by a simple example. Imagine a dataset with two features A and B, with a constraint that $A < B$. Sampling each feature's value independently, as done in LIME's original algorithm, may produce perturbed data points where this constraint is violated.

   LIME may also produce invalid feature values due to its assumption that features are normally-distributed without skews. For instance, let feature A only take positive values with mean $= 1$ and standard deviation $= 2$ (suggesting a right skew). Drawing from a normal distribution with the same mean and standard deviation may result in sampling negative and hence invalid values for A.

   We address this second shortcoming of LIME by implementing a modified algorithm. Specifically, we relax the assumption that features are independent while sampling perturbed points.

6.2.3 Modified Implementation of LIME

LIME's assumption that input features are independent may break down in real-world datasets. As described in Section 4.3.1, the HELOC dataset has 3 pairs of features with correlations greater than 0.8 and thus violates this assumption. While it is possible to remove correlated features with manual feature selection, this undermines one of the strengths of deep learning i.e., the minimal need for feature engineering.

The second shortcoming of LIME described in Section 6.2.2 raises an important question - How does training on perturbed points with possibly invalid values affect the quality of a LIME approximation? The possibility that perturbed data points with invalid values cause a reduction in the performance of the LIME approximations motivates our modified implementation of LIME, in which we relax the assumption of feature independence.

This modification changes the methodology of sampling a data point in Step 3) of the LIME algorithm. In the original implementation, a data point that consists of n features is sampled by sampling each of its n features independently from univariate normal distributions. In our modified implementation, a data point is sampled directly from a joint multivariate normal distribution across all features. This allows the perturbations to be informed by the correlation of features. The multivariate normal distribution is still centered on the mean of each feature value but its standard deviation along each feature is determined by the correlation matrix of the input data.

6.2.4 Evaluating the Original and Modified Implementation of LIME

We use the original and modified implementations of LIME to approximate the best-performing MLP model. We evaluate these approximations on Fidelity and Modification Power in the belief that the implementation whose sample of perturbed points violates fewer constraints will produce a better approximation. As we explore below, this hypothesis is confirmed in certain scenarios. The modified implementation, whose sample of perturbed points contains fewer violations, outperforms the original implementation on Fidelity and Modification Power in certain cases.

Validity of Perturbed Points

As noted, LIME samples perturbed points in the neighborhood of the input data point. We generate a set of $5,000$ perturbed data points from several univariate normal distributions centered at the feature means and a single multivariate normal distribution centered similarly. We measure the quality of the sampled points by two metrics.

1. **Correlation**

   The correlation between features of the perturbed data points ought to be similar to that of the features in the training dataset. We compute the mean squared error (MSE) between the correlation matrix of the training data and that of the perturbed points generated by both implementations of LIME. Let us call these values $MSE^{original}$ and $MSE^{modified}$ respectively. We find that $MSE^{modified}$ is much smaller than $MSE_{original}$ (0.000 vs 0.053). This is expected because the correlation matrix of the training data is an input to the multivariate normal distribution that samples perturbed points in the modified implementation.

2. **Constraint Violations**

   There are 12 constraints relevant to the HELOC dataset (6 relational constraints and 6 value constraints). We derive these manually using domain knowledge with assistance from FICO personnel. The 5,000 perturbed data points are scanned for violations of these constraints.

   One frequently violated constraint is the requirement that all feature values that are interpreted literally and quantitatively (non-special values) must be non-negative. It does not make sense for any of the features in the HELOC dataset to have negative values. For instance, the feature for the number of total lines of credit (NumTotalLOC) cannot be a negative number. We do not consider special values to be negative because they are used symbolically and not interpreted numerically.

   The results for all constraints are shown in Table 6.1 and summarized in Table 6.2. We see that the modified implementation of LIME produces fewer violations than the original implementation in 7 out of the 12 constraints. Moreover, a perturbed point sampled by the modified implementation has on average 1.954 constraint violations vs 2.627 for the original implementation.

Index	Constraint	Original LIME Violations	Modified LIME Violations
1	All feature values interpreted quantitatively must be non-negative	4383	4024
2	PercentLOCNeverDelq $\leq$ 100	1198	1183
3	PercentInstLOC $\leq$ 100	1	2
4	PercentLOCWBalance $\leq$ 100	340	296
5	FracRevLOCLimitUse $\leq$ 100	62	75
6	FracInstLOCUse $\leq$ 100	509	506
7	NumLOC90PlusDaysDelq $\leq$ NumLOC60PlusDaysDelq	1656	655
8	NumLOCReqLast6MExPastWeek $\leq$ NumLOCReqLast6M	2095	388
9	NumLOC60PlusDaysDelq $\leq$ NumTotalLOC	191	239
10	NumLOC90PlusDaysDelq $\leq$ NumTotalLOC	192	233
11	NumLOCNotDelq $\leq$ NumTotal-LOC	2257	1915
12	NumLOCInLast12M $\leq$ NumTo-talLOC	249	255

Table 6.1: Comparison of Constraint Violations Among 5,000 Perturbed Points Sampled from the Original and Modified Implementations of LIME. The points sampled from the modified implementation of LIME have fewer violations in 7 out of the 12 constraints.

Fidelity

We evaluate the Fidelity of approximations produced by the original and modified implementations of LIME with respect to the best-performing MLP. We construct an approximation at each of the $1,973$ points in the test dataset. We compute the fraction of times the MLP and the LIME approximations produce the same classification. We find that the modified implementation achieves a slightly higher Fidelity i.e. $\frac{1937}{1973} = 0.981$ vs the $\frac{1934}{1973} = 0.980$ for those generated by the original implementation. This small difference is not statistically significant.

Original Implementation		Modified Implementation	
Mean	Std. Dev.	Mean	Std. Dev.
2.627	1.323	1.954	1.273

Table 6.2: Count of Constraint Violations by Perturbed Data Generated from Original and Modified Implementations of LIME. The average violations per perturbed point is lower for the modified implementation than that of the original implementation.

Modification Power

We define a modification as a set of minimal changes to a data point intended to reverse its classification. An effective modification is one that produces the desired change in the classification of the data point.

We compute the Modification Power of both implementations of LIME. We use the LIME implementations to generate linear approximations of our MLP at each of the 732 adversely classified points in our test set. Then, we derive modifications for each of these points using the linear approximations. Lastly, we measure the effectiveness of the modifications i.e., the rate with which they are able to reverse the classifications of the adversely classified points.

The coefficients of a traditional linear regression can be used to inform changes to a data point such that it moves from one side of the decision boundary to the other. Similarly, the coefficients of a LIME linear approximation of a black-box machine learning model can be used as to help suggest possible modifications. The key assumption is that the linear approximation will demonstrate high Fidelity to the underlying model both in the neighborhood of the data point in question *and* in the neighborhood of the modified point. If the modifications are small and the linear approximation reverses its classification for the modified point, it is likely the underlying model will also reverse its classification.

We design a generate and test algorithm to find effective modifications that can reverse the classification of a sample data point. First, we generate a possible modification. If the modification is effective, we terminate. Otherwise, we generate another possible modification and repeat the cycle. Once we have exhausted the space of potential solutions, we terminate with a result that no solution was found. We visualize these steps in the flowchart presented in Figure 6-3. We characterize the modifications as domain-agnostic because our algorithm generates them using purely numerical analysis. It does not consider the real-world meaning of the features. We describe what the alternative might entail in Section 7.3.

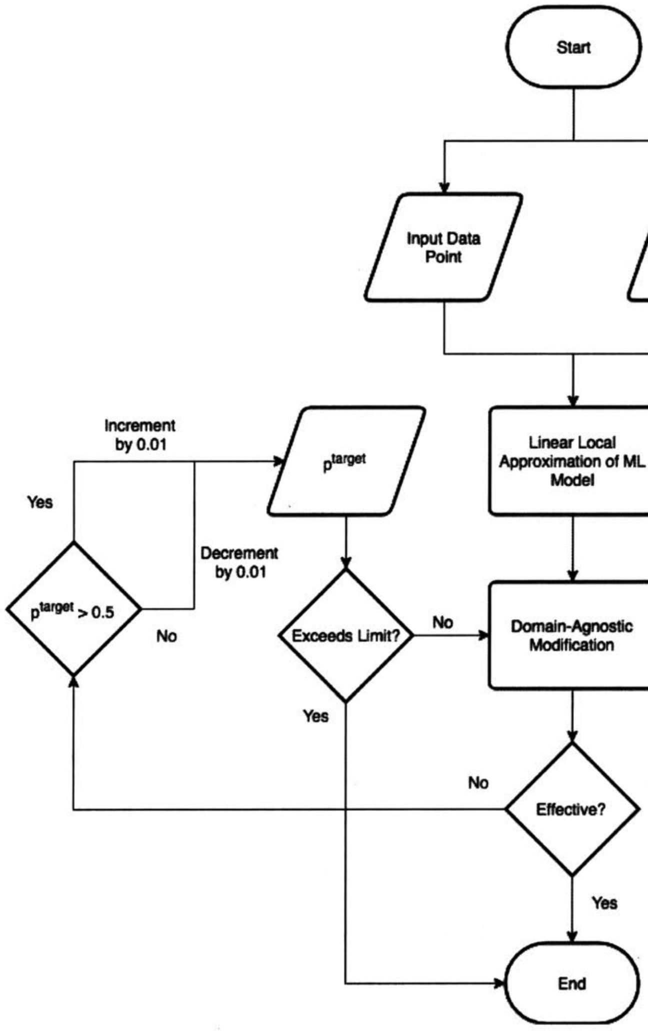

Figure 6-3: Flow Chart for Generating Effective Modifications. Once a modification is generated, we check if it is effective. If yes, we terminate with a solution. Otherwise, we either increase or decrease the value of p^{target} based on the desired label. If p^{target} reaches its limits of 0 or 1 and no effective modification has been found, we terminate with no solution.

Let us now describe our algorithm in detail. Given a machine learning model and a sample data point X whose classification we wish to reverse:

1. A LIME approximation is generated for a trained model at that data point.

2. If we wish to reverse the classification of the data point from adverse to favorable, the probability that it is classified adversely must be less than or equal to 0.5. This can be written formally as $\Pr(Y = 1|X^{modified}) \leq 0.5$. Let us call p^{target} the desired probability for adverse classification. Then, $\Delta p = p^{target} - p^{current}$ is the change in the probability required to reach this target probability. We initialize p^{target} to correspond to the edge of the decision boundary on the side of our desired classification. If the desired classification is favorable, $p^{target} = 0.5$. Otherwise, it is 0.51.

3. We sort the coefficients of the LIME regression, from largest to smallest absolute values in order to produce modifications with the greatest effect per unit change in x_i.

4. In the simplest terms, the values of features are moved up or down to their extreme values until the change that they collectively produce on $\Pr(Y = 1|X)$ is at least equal to the desired Δp.

   Let Δx_i^{min} denote the difference between a feature i's current value and its minimum. Similarly, let Δx_i^{max} denote the difference between the feature's current value and its maximum. For a positive regression coefficient β_i, $\beta_i \cdot \Delta x_i^{min}$ represents the greatest value by which feature i can decrease the output of the regression. Similarly, for a negative β_i, $\beta_i \cdot \Delta x_i^{max}$ represents the greatest value by which feature i can decrease the output of the regression.

   Given the ordering of our features, we choose the smallest k such that:

$$\sum_{i=1}^{k} \beta_i \cdot \Delta x_i^{max/min} \geq \Delta p$$

By modifying features to their extreme values in the order of their corresponding coefficients in the linear approximation, the total Euclidean distance between the original and modified point is minimized. This can be demonstrated by a simple example.

Consider a data point X with two features and assume a linear regression model currently predicts an adverse classification, $Pr(Y = 1|X) = 0.60$ with $\beta_1 = 1$ and $\beta_2 = 0.5$. To correct this classification by decreasing $Pr(Y = 1|X^{modified})$ to 0.40, we need $\Delta p = -0.20$. Two possible modifications are:

(a) Set $\Delta x_1 = -0.20$, producing a change $\beta_1 \cdot \Delta x_1 = -0.20$, the desired amount.

(b) Set $\Delta x_1 = -0.10$ and $\Delta x_2 = -0.20$, which produces a change $\beta_1 \cdot \Delta x_1 + \beta_2 \cdot \Delta x_2 = -0.10 - 0.10 = -0.20$, again the desired amount.

The first modification has a smaller Euclidean distance to the original data point. This is because it only changes the value of x_1, the feature that corresponds to the largest coefficient β_1. This property is desirable because it is more likely that a LIME approximation has a high Fidelity to the underlying model at points closer to the input data point than ones further away.

5. Validate that the LIME approximation predicts the desired classification for the modified data point and determine whether the underlying model predicts the same classification as well.

6. If yes, the algorithm terminates and returns the set of changes as an effective modification. Otherwise, we return to Step 3, decrease or increase p^{target} by 0.01 in the desired direction, and repeat the same steps.

7. If p^{target} has reached its extreme values of 0 or 1 and we have not found an effective modification, we terminate with the result that no effective modification was found.

We run our modification algorithm for all points in the test set that are classified adversely by the MLP i.e., as likely for default. In Table 6.3, we see the effect of varying p^{target} on the rate of successfully reversing the classification of data points from adverse to favorable. By choosing a smaller p^{target}, we try to push a data point further into the favorable region, farther from the decision boundary. As expected, this results in a higher rate of reaching a favorable classification. Interestingly, we see that the modified implementation of LIME has a significantly better Modification Power than that of the original implementation when p^{target} is close to the edge of the decision boundary at 0.5. The significance of this difference decreases as we decrease p^{target}. This makes sense intuitively because coarse approximations may be sufficient to guide a data point deep across the other side of the decision boundary.

p^{target}	Original LIME Modification Power (1)	Modified LIME Modification Power (2)	Difference (2) - (1)
0.5	$\frac{589}{782} = 0.753$	$\frac{636}{782} = 0.813$	$\frac{47}{782} = 0.060$
0.49	$\frac{682}{782} = 0.872$	$\frac{710}{782} = 0.908$	$\frac{28}{782} = 0.036$
0.48	$\frac{747}{782} = 0.955$	$\frac{755}{782} = 0.965$	$\frac{8}{782} = 0.010$
0.47	$\frac{766}{782} = 0.980$	$\frac{768}{782} = 0.982$	$\frac{2}{782} = 0.002$
0.46	$\frac{777}{782} = 0.993$	$\frac{774}{782} = 0.989$	$\frac{-3}{782} = -0.003$
0.45	$\frac{780}{782} = 0.997$	$\frac{777}{782} = 0.993$	$\frac{-3}{782} = -0.003$
0.44	$\frac{782}{782} = 1.000$	$\frac{781}{782} = 0.998$	$\frac{-1}{782} = -0.002$

Table 6.3: Comparison of Modification Power Between Original and Modified LIME. The modified implementation does significantly better when p^{target} is close to the decision boundary. As p^{target} decreases, the difference between the Modification Power of both implementations is not statistically significant.

Our modification algorithm leverages LIME to also be model-agnostic i.e., independent of the model class or implementation. However, its shortcoming is that does not guarantee an effective modification i.e., there may be scenarios in which the modified data point is classified differently by the LIME approximation but its class is unchanged in the original model. This can happen if an approximation demonstrates high Fidelity at the adverse data point but a low Fidelity in the neighborhood of the modified data point. Hence, it is best to minimize the Euclidean distance between the original and modified data point.

Chapter 7

Generating Explanations: Rationales and Effective Modifications

In this chapter, we generate textual explanations for three HELOC applicants using the modified implementation of LIME described in Chapter 6. We provide each applicant rationales for why they were classified the way they were. If they were classified adversely, we suggest actionable steps for them to become Creditworthy and if found Creditworthy, we present cautionary advice for them to remain Creditworthy. Through these examples, we show first that model approximation techniques can be used to generate effective modifications and second that such modifications must be combined with domain-knowledge to produce explanations of pragmatic utility.

7.1 Methodology

We select three HELOC applicants from the test set. Let us denote them as $X^{(1)}$, $X^{(2)}$, and $X^{(3)}$. The first applicant, $X^{(1)}$ and the third, $X^{(3)}$ have ground truth classifications of Non-creditworthy and the second, $X^{(2)}$ as Creditworthy. Our best-performing model, an MLP trained on WoE Data, classifies them correctly as well. The original feature values, the corresponding Weight of Evidence (WoE) bins and WoE values arc tabulated for all three applicants in Tables 7.1, 7.2, and 7.3 respectively.

We compute a linear local approximation of the MLP at each of these sample points using the modified implementation of LIME. We use the regression coefficients of these approximations and the feature values of the data point to identify the factors most influential for each classification. These provide the rationales for the classifications.

As described in Section 6.2.4, our modification algorithm can suggest changes to an input data point and determine if its classification reverses. We use this approach to generate effective modifications for our three sample data points.

7.2 Examples

We use the first two examples to demonstrate how effective modifications can be generated from numerical techniques that are agnostic to the credit domain. The third example serves to show the shortcomings of this approach and motivates the need for incorporating domain-knowledge to produce modifications of pragmatic utility.

7.2.1 Example 1: Actionable Steps For A Non-Creditworthy Applicant

In this example, our algorithm generates an effective modification that reverses the classification of a Non-creditworthy applicant. As described earlier, this modification is generated purely numerically and is agnostic to the domain-specific meaning of our features. We see that the advice suggested is both actionable and practically straightforward to implement for the credit applicant.

Feature	Original Value	WoE Bin	WoE Value
ExternalRiskEstimate	65	[64,68)	0.813
MSinceFirstLOC	172	[138,214)	-0.054
MSinceNewestLOC	4	[2,10):	0.04
AvgAgeOfLOC	68	[60,75)	0.077
NumLOCNotDelq	28	[17,32)	-0.157
NumLOC60PlusDaysDelq	1	[1,2)	0.465
NumLOC90PlusDaysDelq	0	[0,1)	-0.194
PercentLOCNeverDelq	66	[0,82)	1.182
MSinceMRecentDelq	22	[16,32)	0.286
MaxDelqLast12M	G	[G,H)	0.085
MaxDelqEver	E	[C,F)	0.545
NumTotalLOC	29	[28,inf)	-0.264
NumLOCInLast12M	2	[2,3)	0.017
PercentInstLOC	28	[0,29)	-0.24
MSinceNewLOCReqExPastWeek	0	[0,1)	0.371
NumLOCReqLast6M	3	[2,4)	0.211
NumLOCReqLast6MExPastWeek	3	[3,4)	0.25
FracRevLOCLimitUse	45	[38,48)	0.278
FracInstLOCUse	80	[70,84)	0.197
NumRevLOCWBalance	8	[8,inf)	0.584
NumInstLOCWBalance	2	[2,3)	-0.049
NumBank/NatlLoansWHighUtil	1	[1,2)	0.212
PercentLOCWBalance	71	[69,74)	0.069

Table 7.1: Original and WoE Feature Values of The Non-creditworthy Applicant $X^{(1)}$. $MaxDelqEver = E$ means the credit applicant has been 60+ days delinquent ever. $MaxDelqLast12M = G$ means their maximum delinquency in the last 12 months is is unknown.

Classification

The LIME approximation predicts the probability of default of this applicant is 0.805. Figure 7-1 visualizes the contribution of each feature towards this prediction by weighing the regression coefficients of the LIME approximation by the respective WoE encoded feature values. The bars in purple and slanted lines represent the contributions of those features that increase the probability of default and the bars in dotted yellow represent the contribution of the features that decrease that probability. The LIME regression's prediction is the sum of these feature contributions and the regression offset.

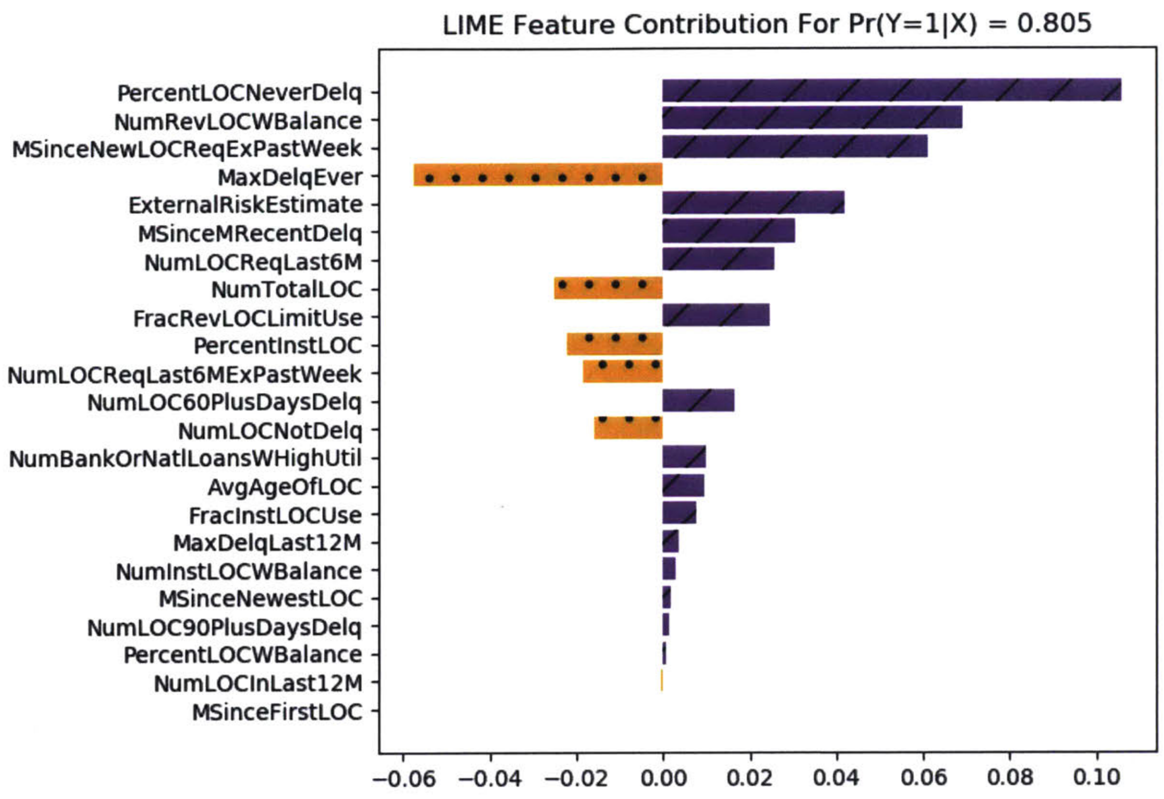

Figure 7-1: Rationales for the Prediction of $\Pr(Y = 1|X^{(1)}) = 0.805$ Produced by the LIME Linear Approximation. The features with bars in purple and slanted lines increased the probability of default and those with bars in dotted yellow decreased that probability.

Using the bars in purple and slanted lines in Figure 7-1, we can identify the top four factors[1] that adversely affected the classification of this applicant. This provides the rationale for the classification.

1. Too many of their lines of credit have been delinquent at least once (PercentLOCNeverDelq).

2. They have too many revolving lines of credit with outstanding balances (NumRevLOCWBalance).

[1]The Fair Credit Reporting Act (FCRA) mandates that a lender must be able to disclose up to four factors that adversely affected a denied applicant

3. They have a poor global credit score (ExternalRiskEstimate).
4. They requested for a new line of credit too recently (MSinceNewLOCReqEx-PastWeek).

Given that X^1 understands the rationale of why they were classified adversely, they would now be interested in actionable steps to reach a favorable classification the next time.

Effective Modifications

An effective modification is one that produces the desired change in the classification of a sample data point. Recall that our algorithm selects an increasingly extreme value of p^{target} in each iteration and finds modifications such that the probability of default of the modified point is equal to p^{target}. In this particular instance, the algorithm finds a modification at $p^{target} = 0.40$ with four changes, all of which need to be made for the applicant to receive a favorable classification:

1. Increase the *number of months since the newest request for a line of credit excluding the past week* from its current value of 0 to 9 or more (MSince-NewLOCReqExPastWeek), and
2. Increase the *average age of all lines of credit* from its current value of 68 to at least 98 (AvgAgeOfLOC), and
3. Decrease the *number of requests for new lines of credit in the last 6 months* from its current value of 3 to 0 (NumLOCReqLast6M), and
4. Decrease the *number of revolving lines of credit with outstanding balances* from its current value of 8 to 5 or fewer (NumRcvLOCWBalance).

Expressed in more straightforward English, the advice suggests:

1. Do not apply for a new line of credit for at least the nine months prior to applying for the HELOC (MSinceNewLOCReqExPastWeek), and

2. Wait for at least 30 months (2.5 years) to improve the average age of your existing lines of credit (AvgAgeOfLOC), and

3. Do not apply for any new lines of credit during the six months prior to applying for the HELOC (NumLOCReqLast6M), and

4. Pay off at least 3 of your 8 revolving lines of credit that have outstanding balances (NumRevLOCWBalance).

Our algorithm determines that these changes do indeed reverse the classification of the Non-Creditworthy applicant to Creditworthy. Note that the advice in Step 3 is a subset of that given in Step 1, though they are motivated by different features (NumLOCReqLast6M and MSinceNewLOCReqExPastWeek respectively). This happens because our algorithm treats these features purely numerically and is agnostic to their meaning. In this particular case, it is not a problem because the applicant can accomplish both steps by completing Step 1.

A linear approximation determines its output $\Pr(Y = 1|X)$ from a linear combination of the X_i's, weighted by the β_i's. As we are interested in moving the applicant to the other side of the decision boundary, we consider the β_i's with the largest absolute values. Changes in the values of the corresponding X_i's will produce the greatest effect on $\Pr(Y = 1|X)$. Notice that these X_i's are not necessarily the same ones that were chosen as key rationales to explain the applicant's adverse classification. For instance, the average age of all lines of credit (AvgAgeOfLOC) and the number of requests for new lines of credit (NumLOCReqLast6M) were not among the four key factors that adversely affected the applicant.

7.2.2 Example 2: Cautionary Advice For A Creditworthy Applicant

Our second example deals with an applicant that was deemed Creditworthy. This applicant would not be interested in understanding how to become Non-creditworthy, but instead in how to remain Creditworthy. We demonstrate how our algorithm can be used to provide cautionary advice that satisfies this need. Again, the effective modification suggested in this example uses a purely numerical analysis and does not incorporate any domain-knowledge.

Feature	Original Value	WoE Bin	WoE Value
ExternalRiskEstimate	82	[79,84)	-1.023
MSinceFirstLOC	374	[325,inf)	-0.661
MSinceNewestLOC	4	[2,10)	0.04
AvgAgeOfLOC	100	[98,inf)	-0.624
NumLOCNotDelq	26	[17,32)	-0.157
NumLOC60PlusDaysDelq	0	[0,1)	-0.28
NumLOC90PlusDaysDelq	0	[0,1)	-0.194
PercentLOCNeverDelq	100	[98,inf)	-0.511
MSinceMRecentDelq	-7	-7	-0.498
MaxDelqLast12M	H	[H,I]	-0.568
MaxDelqEver	8	[H,I]	-0.501
NumTotalLOC	27	[21,28)	-0.068
NumLOCInLast12M	3	[3,4)	0.122
PercentInstLOC	22	[0,29)	-0.24
MSinceNewLOCReqExPastWeek	3	[1,4)	-0.456
NumLOCReqLast6M	2	[2,4)	0.211
NumLOCReqLast6MExPastWeek	2	[2,3)	0.193
FracRevLOCLimitUse	28	[13,29)	-0.353
FracInstLOCUse	94	[84,inf)	0.263
NumRevLOCWBalance	3	[3,4)	-0.237
NumInstLOCWBalance	4	[4,inf)	0.261
NumBank/NatlLoansWHighUtil	2	[2,3)	0.519
PercentLOCWBalance	50	[50,61)	-0.349

Table 7.2: Original and WoE Feature Values of The Creditworthy Applicant $X^{(2)}$. The values of MaxDelqEver and MaxDelqLast12M mean that the applicant has never had a delinquency.

Classification

The LIME approximation of the MLP at $X^{(2)}$ predicts that the probability of default of the applicant is 0.17. This is below the 0.5 threshold and hence, they are Creditworthy. As per the dotted yellow bars in Figure 7-2, we see that the top four factors that contributed favorably towards the applicant's classification are:

1. The average age of their lines of credit is high (AvgAgeOfLOC).
2. They have not applied for new lines of credit too recently (MSinceNewLOCReqExPastWeek).
3. They have not had a delinquency too recently (MSinceMRecentDelq).
4. They have a good global credit score (ExternalRiskEstimate).

Effective Modifications As Cautions

Our algorithm suggests two pieces of cautionary advice at a target probability of default, $p^{target} = 0.51$:

1. Do not decrease the *number of months since the newest request for an additional line of credit* from its current value of 3 to 0 (MSinceNewLOCReqExPastWeek), and
2. Do not decrease the *average age of your lines of credit* from its current value of 100 to less than 28 (AvgAgeOfLOC).

Using the calculations in Appendix 9.1.1, we can restate this advice in more understandable terms as:

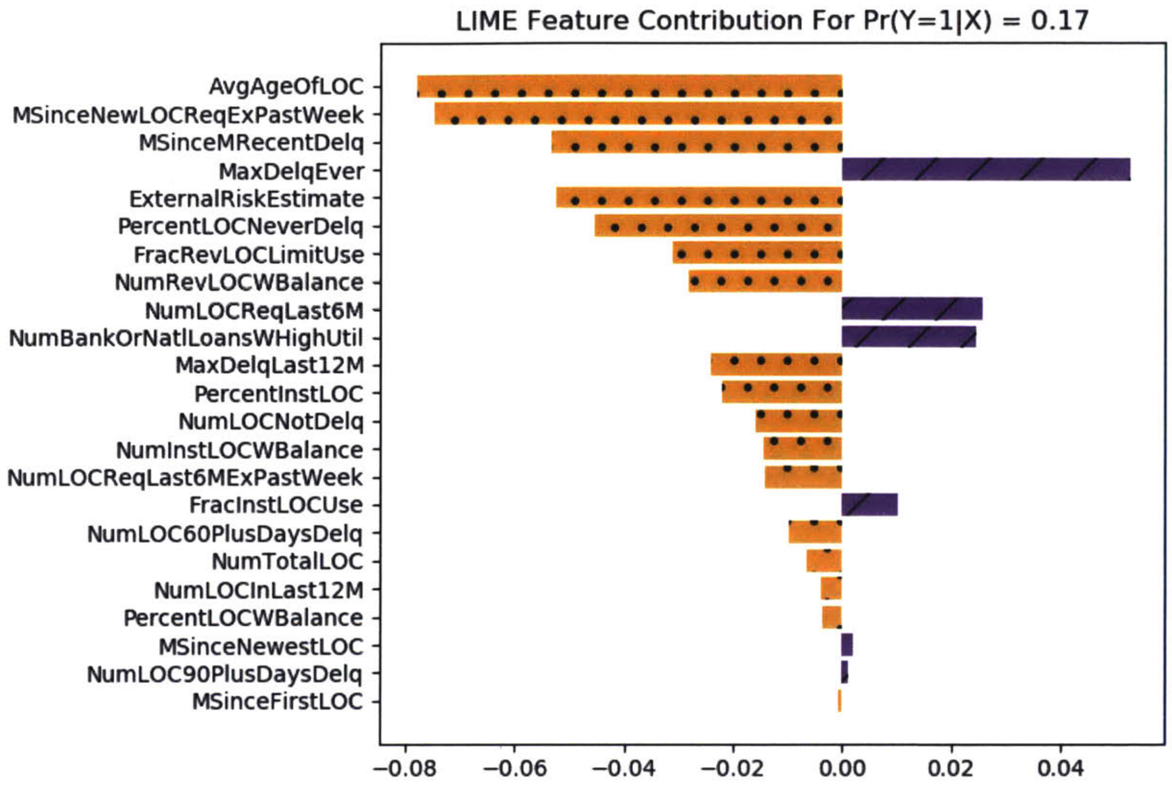

Figure 7-2: Rationales for the Prediction of $\Pr(Y = 1|X^{(2)}) = 0.17$ Produced by the LIME Linear Approximation. The features with bars in purple and slanted lines increased the probability of default and those with bars in dotted yellow decreased that probability.

1. Do not shop for other lines of credit in the month prior to your next HELOC application (MSinceNewLOCReqExPastWeek), and

2. Be wary of opening 70 or more lines of credit because doing so will decrease the average age of your lines of credit (AvgAgeOfLOC) too substantially.

The second piece of advice might sound extreme but it informs the candidate that they can comfortably open a few additional lines of credit, should they desire.

7.2.3 Example 3: Actionable Steps For A Non-Creditworthy Applicant, A Problematic Case

The fact that we are generating modifications agnostic to the credit domain will become particularly evident in this example. In contrast to the previous examples, we see that our algorithm generates a modification that is not pragmatically-useful. To address this, we augment our algorithm with domain-knowledge and demonstrate how we can generate an effective and domain-informed modification. An interesting aspect about the modification in this examples is that it advises the applicant both on what they should do and what they shouldn't, to reach a favorable classification.

Feature	Original Value	WoE Bin	WoE Value
ExternalRiskEstimate	62	[60,64)	1.25
MSinceFirstLOC	275	[214,325)	-0.342
MSinceNewestLOC	6	[2,10)	0.04
AvgAgeOfLOC	99	[98,inf)	-0.624
NumLOCNotDelq	21	[17,32)	-0.157
NumLOC60PlusDaysDelq	1	[1,2)	0.465
NumLOC90PlusDaysDelq	1	[1,2)	0.536
PercentLOCNeverDelq	91	[89,93)	0.365
MSinceMRecentDelq	12	[9,16)	0.694
MaxDelqLast12M	G	[G,H)	0.085
MaxDelqEver	G	[G,H)	0.308
NumTotalLOC	22	[21,28)	-0.068
NumLOCInLast12M	1	[1,2)	-0.096
PercentInstLOC	55	[48,79)	0.471
MSinceNewLOCReqExPastWeek	15	[9,inf)	-0.9
NumLOCReqLast6M	0	[0,1)	-0.302
NumLOCReqLast6MExPastWeek	0	[0,1)	-0.284
FracRevLOCLimitUse	87	[77,inf)	1.451
FracInstLOCUse	71	[70,84)	0.197
NumRevLOCWBalance	6	[6,8)	0.286
NumInstLOCWBalance	5	[4,inf)	0.261
NumBank/NatlLoansWHighUtil	5	[5,inf)	1.207
PercentLOCWBalance	92	[87,inf)	0.914

Table 7.3: Original and WoE Value Of The Non-Creditworthy Applicant. $MaxDelqEver = G$ and $MaxDelqLast12M = G$ both means it is not known if the applicant has ever had a delinquency in the respective time lines.

Classification

The LIME approximation of the MLP predicts the probability of default of this applicant is 0.695. This is above the 0.5 threshold and matches the applicant's ground-truth label of Non-creditworthy. Using Figure 7-3, we identify the rationales for the applicant's adverse classification:

1. They use a high fraction of their available revolving credit limits (FracRevLOCLimitUse).
2. They had a delinquency too recently (MSinceMRecentDelq).
3. They have a poor global credit score (ExternalRiskEstimate).
4. The have too many bank or national loans with high utilization (NumBank/NatLoansWHighUtil.)

Effective Modifications

At the very first value of $p^{target} = 0.5$, the algorithm finds an effective modification with six suggestions. The applicant must act appropriately on all of them to be deemed Creditworthy.

1. Do not decrease the *number of months since the newest request for a line of credit* from its desirable current value of 15 (MSinceNewLOCReqExPastWeek), and
2. Do not decrease the *average age of all lines of credit* from its desirable current value of 99 (AvgAgeOfLOC), and
3. Do not increase the *number of requests for new lines of credit in the past 6 months* from its desirable current value of 0 (NumLOCReqLast6M), and

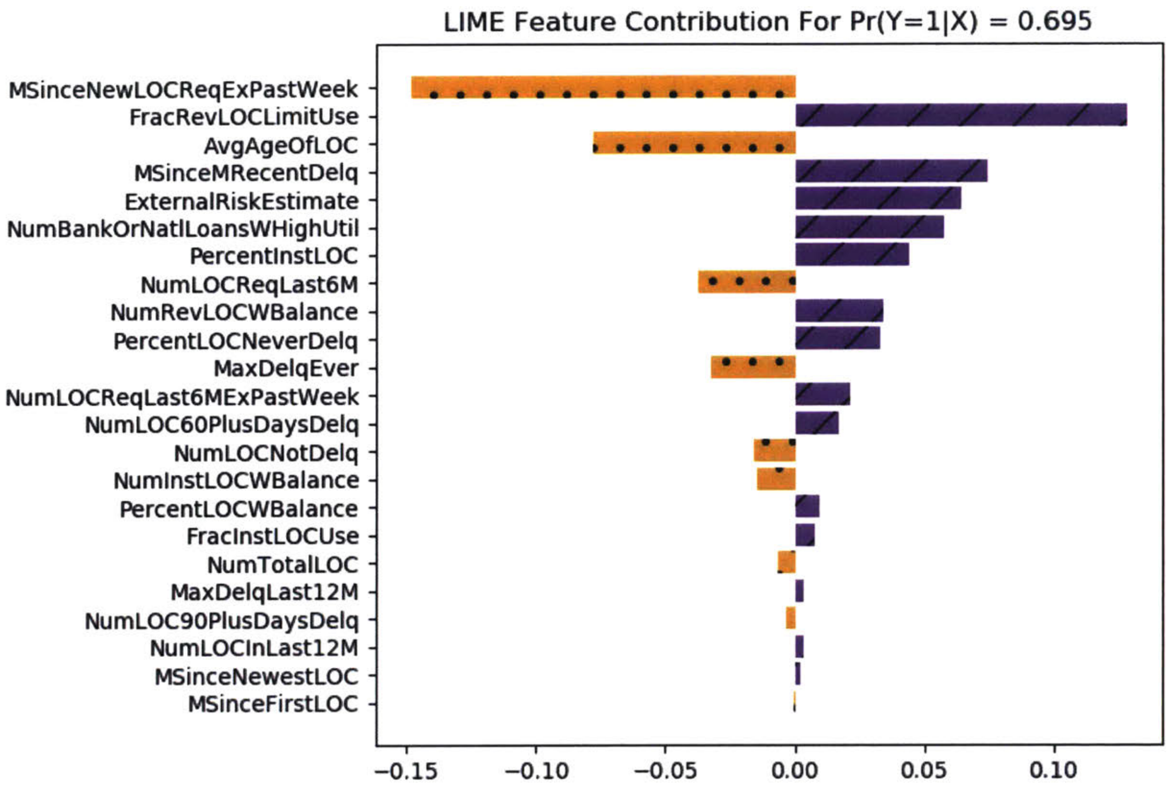

Figure 7-3: Rationales for the Prediction of $\Pr(Y = 1|X^{(3)}) = 0.695$ Produced by the LIME Linear Approximation. The features with bars in purple and slanted lines increased the probability of default and those with bars in dotted yellow decreased that probability.

4. Decrease the *number of revolving loans with outstanding balances* from its current value of 6 to 2 or fewer. (NumRevLOCWBalance), and

5. Increase the *number of months since the most recent delinquency* from its current value of 12 to 48 or more (MSinceMRecentDelq), and

6. Change the *maximum delinquency ever* from its current value of 30+ days delinquent (MaxDelqEver)

This advice reveals both a noteworthy capability and a shortcoming of the modification algorithm. While Section 7.2.1 shows how the algorithm can advise a favorably classified applicant on what not to do, this example shows its capability to do the same for an adversely classified applicant. In fact, the first three modifications are instances of cautionary pieces of advice. The caveat is that the algorithm has identified only those features with the will produce the highest change in output per unit change in feature values. This list is thus not exhaustive.

The shortcoming of the algorithm is that in this particular instance, one of its suggested modifications is infeasible: The applicant cannot change the value of their maximum delinquency ever (MaxDelqEver) to a less unfavorable value. Once an applicant has been 30+ days delinquent *ever*, they can only further deteriorate the feature value by being more delinquent (say by 60+ days) but they cannot improve it. This shortcoming cannot be addressed if the algorithm is domain-agnostic. Instead, it must use domain-knowledge to distinguish between features that are controllable by the applicant and those that are not. We implement this change and run the modification algorithm on the same applicant. It arrives at a modification with identical changes in Steps 1 to 5, but this time suggests modifications to the features NumLOCNotDelq and NumTotalLOC in place of MaxDelqEver in Step 6. These additional steps are:

1. Increase the *number of lines of credit that are not currently delinquent* from its current value of 21 to at least 32 (NumLOCNotDelq), and
2. Increase the *number of total lines of credit* from 22 to at least 28 (NumTotalLOC).

Once again, expressed in more straightforward English, the advice suggests:

1. Do not apply for a new line of credit prior to applying for the HELOC (MSinceNewLOCReqExPastWeek), and
2. Do not apply for any new line of credit for the six months prior to applying for the HELOC (NumLOCReqLast6M), and
3. Pay off at least 4 of your 6 revolving lines of credit that have outstanding balances (NumRevLOCWBalance), and

4. Do not have another delinquency for the next 3 years (MSinceMRecentDelq), and

5. Open 11 new lines of credit that are not delinquent (NumLOCNotDelq), and

6. Open 6 additional lines of credit (NumTotalLOC).

Among other actions, the advice suggests the credit applicant open 11 additional lines of credit. Given that the applicant already has 22 lines of credit, this may be an infeasible request. Embarking on this advice may prove to be particularly problematic for the applicant in case they do not get approved for all 11 lines of credit. Yet, even if they did, it would result in second-order effects on the values of other features. As per the calculations in Appendix 9.2, there are six additional effects that we must account for:

1. The number of months since the applicant opened their newest line of credit will drop from 6 to 0 (MSinceNewestLOC).

2. Immediately after all new lines of credit have been established, the average age of all lines of credit will reduce from 99 to 66 (AvgAgeOfLOC).

3. The percentage of lines of credit that have never been delinquent will increase from 91 to 94 (PercentLOCNeverDelq).

4. The number of lines of credit the applicant will have opened in the last 12 months will increase from 1 to 12 (NumLOCInLast12M).

5. Assuming all new lines of credit are revolving, the percentage of installment lines of credit will decrease from 55 to 37 (PercentInstLOC).

6. Assuming the applicant has not yet started using the newly established lines of credit, the percent of lines of credit with outstanding balances will decrease from 92 to 61 (PercentLOCWBalance).

Additional complications arise because some of the second-order effects are complex.

1. The new external risk estimate cannot be computed without knowing its formula (ExternalRiskEstimate).
2. The new ratio of the fraction of the revolving lines of credit used will depend on the credit limits of the new lines of credit (FracRevLOCLimitUse).
3. The number of lines of credit the candidate applied for in the previous six months depends on whether they applied for all 11 credit lines at once, staggered them over a period of time, or were offered to open some of the new credit lines, in which case they do not count towards their requests for lines of credit (NumLOCReqLast6M, NumLOCReqLast6MExPastWeek).

The explanation we provide above is both effective and consists only of changes that a credit applicant can control. In particular, it avoids its previous mistake of recommending changes to the external risk estimate, a feature that a credit applicant cannot directly control. We generate this explanation using our modification algorithm by explicitly defining which features are controllable and which are not. However, the explanation falls short in asking a credit applicant to modify a controllable feature by a potentially infeasible value: opening 11 additional lines of credit may not be practical. Furthermore, the modification implicitly assumes independence of features, yet opening 11 new lines of credit will result in second-order effects on six additional feature values, some precisely measurable and others uncertain. In fact, once we account for these second-order effects on the feature values of the modified point, it is classified again as Non-creditworthy by the MLP model. Simply put, the modification fails.

Clearly, it is insufficient to consider only the numerical properties of features to generate pragmatically-useful modifications. We must also understand whether features are controllable, and if so, what changes to their feature values are feasible.

We need to go even further and consider the interdependencies between features, so that if one feature value changes, we can account for its indirect effects on other feature values.

7.3 Explanatory Utility

The examples in this chapter illustrate that treating features purely numerically, agnostic to their domain-specific meaning, can produce effective modifications that may not be pragmatically-useful. The modifications could have three types of shortcomings:

1. They suggest changes to the value of a feature that is in fact not controllable by a credit applicant.
2. They suggest practically infeasible changes to the value of a feature, even if it is controllable.
3. They do not account for the second-order effects of changing feature values, resulting from interdependencies among features.

To deal with these, we claim that our algorithm must consider the *explanatory utility* of each feature in a modification. We define explanatory utility as the pragmatic utility of a feature in an explanation. Such a value depends not only on the feature itself but also on the change in its value proposed. For instance, a controllable feature like the total number of lines of credit (NumTotalLOC) has a higher explanatory utility than an uncontrollable one such as the external risk estimate (ExternalRiskEstimate). Within a controllable feature like NumTotalLOC, increasing the number of lines of credit from 0 to 11 is more practical and thus has a higher explanatory utility than increasing it from 22 to 33. Developing such a notion requires domain expertise and human judgment.

Our algorithm described in Section 6.2.4 generates *domain-agnostic* modifications. To generate *domain-informed* modifications, we augment our algorithm with domain-knowledge in three ways:

1. **Account for the Controllability of Features** - We explicitly categorize each predictor feature as controllable (e.g. the number of total lines of credit) or non-controllable (e.g. the external risk estimate). We constrain our algorithm to suggest changes only to the values of features that are controllable.

2. **Account for Second-Order Effects** - We consider the interdependencies between features and account for them programmatically. For instance, if a new line of credit is opened, the average age of all lines of credit will change in a computationally predictable manner, among other effects. The effects of some features may not be as precise and require assumptions about the future behavior of a credit applicant (e.g. what is the most likely effect of opening a new line of credit on the fraction of the revolving lines of credit used?). These must be constructed in consultation with domain experts.

3. **Evaluate the Feasibility of Changing Feature Values** - We use human judgment to determine if an effective and domain-informed modification is also pragmatically-useful. We believe that there is an opportunity to automate this step in future work.

These changes are reflected in the flowchart presented in Figure 7-4. Once a *domain-agnostic modification* is generated, we determine whether it is effective. If yes, we use domain-knowledge to constrain the modification to changes in controllable features only and account for second-order effects due to interdependencies among features. We then produce a *domain-informed modification.* We test for its effectiveness programmatically and finally, use human judgment to evaluate its pragmatic utility.

In summary, we demonstrate that generating explanations for a credit applicant should not be a purely numerical exercise. Domain-knowledge can and must be incorporated to generate pragmatically-useful modifications.

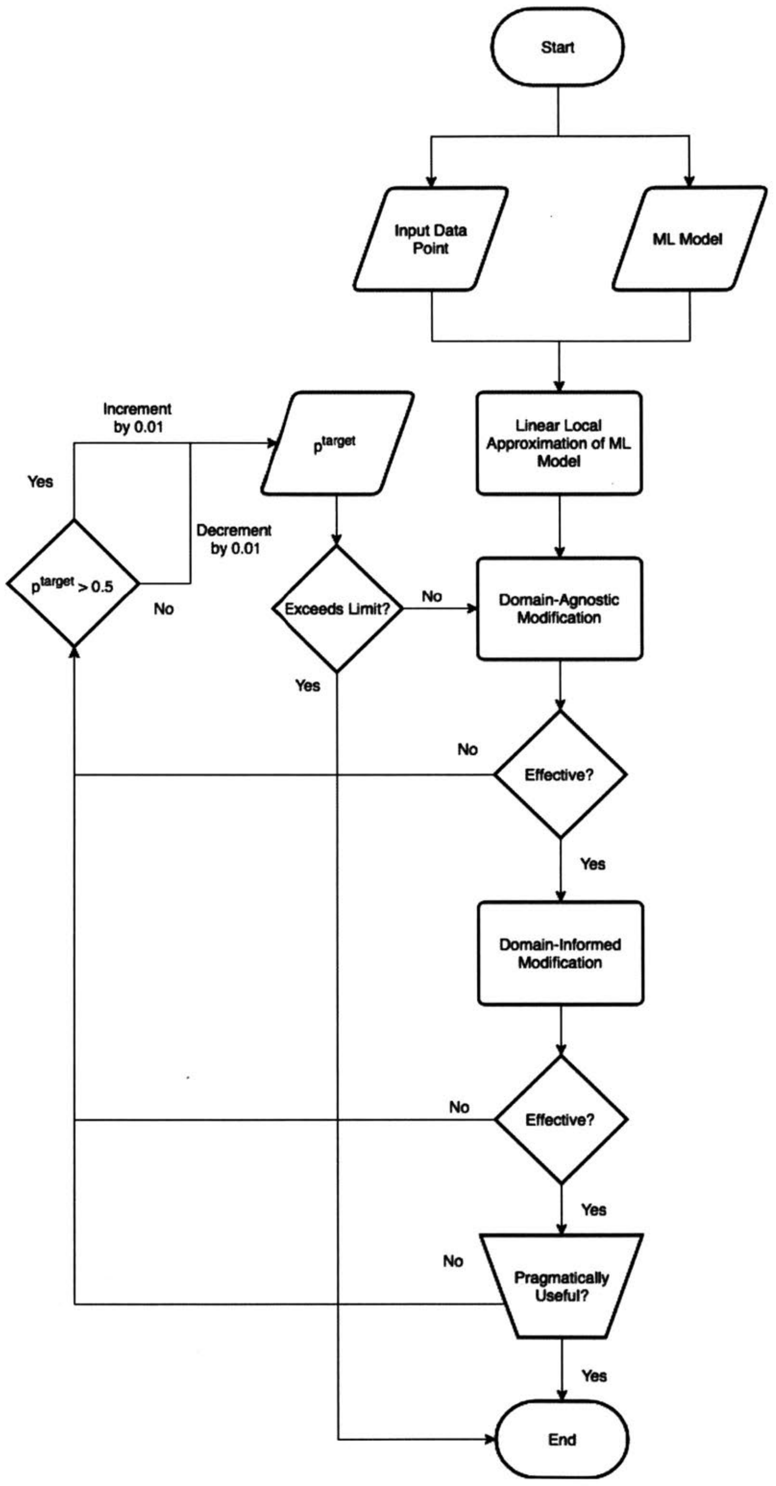

Figure 7-4: Flow Chart for Generating Domain-Informed Modifications. Once it is determined that a domain-agnostic modification is effective, the explanatory utility of features are considered to produce a domain-informed modification. If found effective as well, the modification is evaluated by human judgment for its pragmatic utility.

Chapter 8

Conclusion and Next Steps

This thesis focused on answering two questions relevant to understanding the prediction of any machine learning model for a sample data point. Why was the data point classified the way it was? What small changes in its feature values could reverse its classification? We motivated these objectives by the needs of the credit risk modeling domain. Specifically, we chose to generate explanations most useful for a credit applicant. These include *rationales* for their classification. They also include *effective modifications* that suggest actionable steps to reverse their classification. While the former is a legal requirement by the Fair Credit Report Act, the latter is of considerable practical use to a denied credit applicant. The key takeaway of the thesis is that while model-approximation techniques can be used to generate effective modifications, they must incorporate domain-knowledge to produce modifications of pragmatic utility.

8.1 Key Results

In this thesis, we considered simplicity as a key ingredient in interpretability. This motivated our approach to interpreting a black-box model by approximating it with a simpler model. We considered linear regressions to be simple because of their decomposability into individual coefficients. Instead of approximating an entire black-box model with a single, poorly-performing linear approximation, we chose to approximate the model piece-wise using several linear approximations. We constructed local approximations of our best-performing MLP using LIME, a local model approximation technique. We then used these approximations to generate *rationales* and *effective justifications* for credit applicants.

LIME approximates the decision boundary of a black-box machine learning model at a particular data point by constructing a locally-weighted linear regression. The regression is trained on perturbed data points in the neighborhood of the input data point. The original implementation of LIME assumed that features are independent. As described in Section 4.3.1, our HELOC dataset violated this assumption as it had 3 pairs of features with correlations greater than 0.8. As a result, sampling feature values independently, as per the original implementation of LIME, resulted in sampling invalid perturbed data points. To address this shortcoming, we implemented a modified version of LIME that relaxed this assumption and accounted for the correlation of features. We found that the correlation of features of perturbed points sampled from our modified implementation resembled that of the training data more closely compared to that of perturbed points sampled from the original implementation. We also found that the modified implementation of LIME sampled perturbed points with greater validity, as measured by the number of constraints violated.

We evaluated the two implementations of LIME on their ability to approximate our best-performing model - a Multi-Layer Perceptron (MLP) trained on WoE data with 74.7% test accuracy. The two criteria for the evaluation were Fidelity and Modification Power. Fidelity measured how often the linear approximation matched the prediction of the underlying model at the data point of interest. Modification Power measured the rate with which the classification of a set of adversely classified data points could be reversed using modifications derived from linear approximations. We found that the modified implementation of LIME outperformed the original implementation in both criteria in certain scenarios. While this was not true in all scenarios, we believe that accounting for the correlations of features in LIME's algorithm, as we did, is the correct approach when dealing with datasets with highly correlated features.

Using a linear approximation as a proxy for the decision boundary of an underlying model, we computed *rationales* for predictions. We also used the linear approximations to implement a generate and test algorithm that can suggest *effective modifications* intended to reverse the classification of a data point. Our algorithm identifies changes to the feature values of a sample credit applicant to achieve a target probability of default p^{target}. It also produces changes that result in the smallest possible Euclidean distance between the original and modified point. It accomplishes this by optimizing changes in the features with the greatest coefficient values. An interesting result was that features that might contribute most adversely to a classification may not be the best features to change in a modification.

Although our methodology does not guarantee an effective modification, we saw promising results empirically on our HELOC dataset. As per Table 6.3, we found that the effectiveness of modifications increased with increasingly extreme (lower in this case) values for p^{target}. We also saw that the modified implementation of LIME had a significantly greater Modification Power compared to the original implementation when p^{target} was close to the edge of the decision boundary. The significance of this difference decreased as we decreased p^{target}. This result made sense intuitively because even coarse approximations may be sufficient to guide a data point deep across the other side of the decision boundary.

We used three examples to show that effective modifications may not be pragmatically useful. The modifications may suffer from three possible types of shortcomings:

1. They suggest changes to the value of a feature that is in fact not controllable by a credit applicant.

2. They suggest practically infeasible changes to the value of a feature, even if it is controllable.

3. They do not account for the second-order effects of changing feature values, resulting from interdependencies among features.

Through our results, we claim that these shortcomings cannot be dealt with by numerical analysis that is agnostic to the domain of the data. Instead, we argue that modifications must use domain-knowledge and consider the explanatory utility of each feature i.e., its pragmatic utility in an explanation. We augment our domain-agnostic modification algorithm to generate *domain-informed* modifications in three ways:

1. **Account for the Controllability of Features** - We explicitly categorize each predictor feature as controllable (e.g. the number of total lines of credit) or non-controllable (e.g. the external risk estimate). We constrain our algorithm to suggest changes only to the values of features that are controllable.

2. **Account for Second-Order Effects** - We consider the interdependencies between features and account for them programmatically. For instance, if a new line of credit is opened, the average age of all lines of credit will change in a computationally predictable manner, among other effects.

3. **Evaluate the Feasibility of Changing Feature Values** - We use human judgment to determine if an effective and domain-informed modification is also pragmatically-useful.

8.2 Further Work

There are several possible directions to further this work. As we outlined in Section 6.2.2, the coefficients of a LIME regression cannot be interpreted reliably in the presence of highly correlated features. As a consequence, the rationales produced by LIME regressions are not guaranteed to be reliable as well. SHAP is another local interpretability technique described in Section 3.2.1 that can produce feature attributions for a particular prediction, robust to the presence of highly correlated features. However, unlike LIME, SHAP does not produce a linear approximation to the underlying model. Due to this, we cannot directly use SHAP to generate effective modifications using our generate and test algorithm. One potential solution might be to infer a local approximation that is consistent with the SHAP feature attributions for a particular data point and the feature values of that data point. It would be interesting to benchmark the performance of a SHAP-inferred approximation with a LIME approximation on our criteria of evaluation, namely Fidelity and Modification Power.

Second, we modified the implementation of LIME to relax the assumption of independence of features in sampling perturbed points. We found that the modified implementation of LIME had several desirable properties. It sampled perturbed points with greater validity. The correlation of features in the perturbed points sampled from the modified implementation closely resembled that of features in the training data. Moreover, the average violation of constraints per perturbed point sampled was lower for the modified implementation than that of the original implementation. In certain scenarios, we also saw the modified implementation of LIME outperformed the original implementation, as measured by Fidelity and Modification Power.

One limitation of our modified implementation is that it only accounts for the correlation of continuous features while sampling perturbed points. As a result, one extension of our work could be to account for the correlations of categorical features while sampling perturbed points.

Lastly and most importantly, we believe there is a scope to improve our generate and test modification algorithm. In this thesis, we demonstrate that purely numerical analysis, agnostic to the domain of the data cannot produce modifications of pragmatic utility. By augmenting our modification algorithm, we demonstrate how domain-knowledge can be incorporated to produce effective modifications that are also pragmatically-useful. We accomplished this by considering the explanatory utility of features: whether they are controllable, what changes in their feature values are feasible, and how they are interdependent with each other.

In the final step of our correction algorithm, we use human judgment to evaluate the practical feasibility of the changes in feature values suggested by an effective and domain-informed modification. One extension of our work is to automate this judgment. For instance, could our algorithm automatically decide that it is better to advise an applicant to open a smaller number of lines of credit than to wait for an unreasonably long period of time to improve the average age of their lines of credit? Such an exercise could itself be a machine learning task that learns from previous human judgments about the explanatory utility of suggested changes.

Another extension to our work could be to account for second-order effects with greater granularity. In our work, we demonstrate that opening a new line of credit affects the values of several interdependent features. Estimating the precise change in the values of some features requires the ability to predict the future behavior of credit applicants. For instance, how does the fraction of the revolving credit limit used by a credit applicant tend to change after they have opened one new line of credit or ten? We demonstrated in this work that capturing such second-order effects as accurately as possible is essential for generating explanations of pragmatic utility.

Chapter 9

Appendix

9.1 Calculations for Example 2

In this section, we include calculations required for Example 2 in Chapter 7.

9.1.1 Average Age of All Lines Of Credit

We are interested in computing the number of additional lines of credit that would have to be opened to change the average age of all lines of credit from its current value of 100 to 28.

We know the formula,

$$AvgAgeOfLOC = \frac{TotalAgeOfLOC}{NumTotalLOC}$$

Or,

$$TotalAgeOfLOC = AvgAgeOfLOC \cdot NumTotalLOC = 100 \cdot 27 = 2700$$

Months.

Our target value for

$$AvgAgeOfLOC^{modified} = 28$$

Hence,

$$NumTotalLOC^{modified} = \frac{TotalAgeOfLOC}{AvgAgeOfLOC^{modified}} = \frac{2700}{28} \approx 96$$

Therefore,

$$\Delta NumTotalLOC = NumTotalLOC^{modified} - NumTotalLOC = 96 - 26 = 70$$

9.2 Calculations for Example 3

In this section, we detail the calculations for Example 3 in Chapter 7. Specifically, we are interested in computing the second order effects of increasing the total number of lines of credit from 22 to 33 by opening 11 new revolving lines of credit.

9.2.1 Average Age of All Lines Of Credit

We find that $AvgAgeOfLOC^{modified} = 66$.

We know that,

$$AvgAgeOfLOC = \frac{TotalAgeOfLOC}{NumTotalLOC}$$

Or,

$$TotalAgeOfLOC = AvgAgeOfLOC \cdot NumTotalLOC = 99 \cdot 22 = 2178$$

Months.

Once we open the 11 revolving lines of credit,

$$NumTotalLOC^{modified} = 33$$

Hence,

$$AvgAgeOfLOC^{modified} = \frac{TotalAgeOfLOC}{NumTotalLOC} = \frac{2178}{33} = 66$$

9.2.2 Percentage Of Lines Of Credit Never Delinquent

We compute that $PercentLOCNeverDelq^{modified} = 94$.

By definition,

$$PercentLOCEverDelq = \frac{100 - PercentLOCNeverDelq}{100} = 0.09$$

The formula for

$$PercentLOCEverDelq = \frac{NumLOCEverDelq}{NumTotalLOC}$$

Hence,

$$NumLOCEverDelq = PercentLOCEverDelq \cdot NumTotalLOC = 0.09 \cdot 22 \approx 2$$

Given that

$$NumTotalLOC^{modified} = 33$$

We get

$$PercentLOCEverDelq^{modified} = \frac{NumLOCEverDelq}{NumTotalLOC^{modified}} = \frac{2}{33} \approx 0.06$$

Or,

$$PercentLOCNeverDelq^{modified} = (1-PercentLOCEverDelq^{modified})\cdot 100 = (1-0.06)\cdot 100 = 94$$

9.2.3 Percentage Of Installment Lines Of Credit

We calculate that $PercentInstLOC^{modified} = 37$

By definition,

$$PercentInstLOC = \frac{NumInstLOC}{NumTotalLOC} \cdot 100$$

Rearranging,

$$NumInstLOC = \frac{PercentInstLOC \cdot NumTotalLOC}{100} = \frac{55 \cdot 22}{100} = \frac{1210}{100} \approx 12$$

By opening 11 new revolving lines of credit, the number of installment lines of credit remains unchanged.

Plugging in

$$NumTotalLOC^{modified} = 33$$

We see that

$$PercentInstLOC^{modified} = \frac{NumInstLOC}{NumTotalLOC^{modified}} \cdot 100 = \frac{12}{33} \cdot 100 \approx 37$$

9.2.4 Percentage Of Lines Of Credit With Balance

We calculate that $PercentLOCWBalance^{modified} = 61$

By definition,

$$PercentLOCWBalance = \frac{NumLOCWBalance}{NumTotalLOC} \cdot 100$$

Rearranging,

$$NumLOCWBalance = \frac{PercentLOCWBalance \cdot NumTotalLOC}{100} = \frac{92 \cdot 22}{100} = \frac{2024}{100} \approx 20$$

By opening 11 new revolving lines of credit, the number of lines of credit remains unchanged.

Using

$$NumTotalLOC^{modified} = 33$$

We find that

$$PercentLOCWBalance^{modified} = \frac{NumLOCWBalance}{NumTotalLOC^{modified}} \cdot 100 = \frac{20}{33} \cdot 100 \approx 61$$

Bibliography

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances In Neural Information Processing Systems* (2012). ISSN: 10495258. DOI: `http://dx.doi.org/10.1016/j.protcy.2014.09.007`. arXiv: `1102.0183`.

[2] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *International Conference on Learning Representations (ICRL)* (2015). ISSN: 09505849. DOI: `10.1016/j.infsof.2008.09.005`. arXiv: `1409.1556`.

[3] *The Center for Microeconomic Data.* 2018. URL: `https://www.newyorkfed.org/microeconomics/hhdc.html` (visited on 07/31/2018).

[4] Andrew Ng. *Machine Learning.* URL: `https://www.coursera.org/learn/machine-learning` (visited on 07/31/2018).

[5] Nykamp DQ. *Function definition - Math Insight.* URL: `https://mathinsight.org/definition/function` (visited on 07/31/2018).

[6] Justin Sirignano, Apaar Sadhwani, and Kay Giesecke. “Deep Learning for Mortgage Risk”. In: (July 2016). arXiv: `1607.02470`. URL: `http://arxiv.org/abs/1607.02470`.

[7] Bryce Goodman and Seth Flaxman. “European Union regulations on algorithmic decision-making and a "right to explanation"”. In: (2016). DOI: `10.1609/`

aimag.v38i3.2741. arXiv: 1606.08813. URL: http://arxiv.org/abs/1606.08813%20http://dx.doi.org/10.1609/aimag.v38i3.2741.

[8] Finale Doshi-Velez and Been Kim. "Towards A Rigorous Science of Interpretable Machine Learning". In: (Feb. 2017). arXiv: 1702.08608. URL: http://arxiv.org/abs/1702.08608.

[9] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?: Explaining the Predictions of Any Classifier". In: (Feb. 2016). arXiv: 1602.04938. URL: http://arxiv.org/abs/1602.04938.

[10] Cody Marie Wild. *One Feature Attribution Method to (Supposedly) Rule Them All: Shapley Values.* URL: https://towardsdatascience.com/one-feature-attribution-method-to-supposedly-rule-them-all-shapley-values-f3e04534983d (visited on 07/31/2018).

[11] Scott Lundberg and Su-In Lee. "An unexpected unity among methods for interpreting model predictions". In: (Nov. 2016). arXiv: 1611.07478. URL: http://arxiv.org/abs/1611.07478.

[12] David Baehrens et al. "How to Explain Individual Classification Decisions". In: (Dec. 2009). arXiv: 0912.1128. URL: http://arxiv.org/abs/0912.1128.

[13] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic Attribution for Deep Networks". In: (Mar. 2017). arXiv: 1703.01365. URL: http://arxiv.org/abs/1703.01365.

[14] Xin Zhang, Armando Solar-Lezama, and Rishabh Singh. "Interpreting Neural Network Judgments via Minimal, Stable, and Symbolic Corrections". In: (Feb. 2018). arXiv: 1802.07384. URL: http://arxiv.org/abs/1802.07384.

[15] R. Dennis. Cook and Sanford Weisberg. *Residuals and influence in regression.* Chapman and Hall, 1982, p. 230. ISBN: 0412242800. URL: https://conservancy.umn.edu/handle/11299/37076.

[16] Pang Wei Koh and Percy Liang. "Understanding Black-box Predictions via Influence Functions". In: (Mar. 2017). arXiv: 1703.04730. URL: http://arxiv.org/abs/1703.04730.

[17] Been Kim Google Brain. *Interpretable Machine Learning: The fuss, the concrete and the questions.* Tech. rep. URL: https://xkcd.com/.

[18] Joseph Sill. *Monotonic Networks.* Tech. rep. URL: https://papers.nips.cc/paper/1358-monotonic-networks.pdf.

[19] Maya Gupta, Jan Pfeiffer, and Seungil You. *TensorFlow Lattice: Flexibility Empowered by Prior Knowledge.* 2017. URL: https://ai.googleblog.com/2017/10/tensorflow-lattice-flexibility.html (visited on 07/31/2018).

[20] *Introducing TensorFlow Feature Columns.* 2017. URL: https://developers.googleblog.com/2017/11/introducing-tensorflow-feature-columns.html (visited on 07/31/2018).

[21] Dhruv Sharma. "Improving the Art, Craft and Science of Economic Credit Risk Scorecards Using Random Forests: Why Credit Scorers and Economists Should Use Random Forests". In: *SSRN Electronic Journal* (June 2011). ISSN: 1556-5068. DOI: 10.2139/ssrn.1861535. URL: http://www.ssrn.com/abstract=1861535.

[22] Gilles Louppe. "Understanding Random Forests: From Theory to Practice". In: (July 2014). arXiv: 1407.7502. URL: http://arxiv.org/abs/1407.7502.

[23] Kanchan Sarkar. *ReLU : Not a Differentiable Function: Why used in Gradient Based Optimization?* 2018. URL: https://medium.com/@kanchansarkar/relu-not-a-differentiable-function-why-used-in-gradient-based-optimization-7fef3a4cecec (visited on 07/31/2018).

[24] Zachary C. Lipton. "The Mythos of Model Interpretability". In: (June 2016). arXiv: 1606.03490. URL: http://arxiv.org/abs/1606.03490.